VAX/VMS Release Notes, Version 4.5

Order Number: AA–JF88A–TE

September 1986

This document describes Version 4.5 of the VAX/VMS operating system and explains the method for updating a Version 4.4 system to Version 4.5. It lists and discusses system changes, new features, corrected problems, and restrictions in the use of the system. It also describes changes and corrections to the VAX/VMS documentation set.

Revision/Update Information:	This is a new manual, which adds to and corrects information contained in the *VAX/VMS Release Notes, Version 4.4*, Order Number AA–Z107A–TE, published April, 1986.
Operating System and Version:	VAX/VMS Version 4.5
Software Version:	VAX/VMS Version 4.5

digital equipment corporation
maynard, massachusetts

September 1986

The postpaid READER'S COMMENTS form on the last page of this document requests the user's critical evaluation to assist in preparing future documentation.

The following are trademarks of Digital Equipment Corporation:

DEC	DIBOL	UNIBUS
DEC/CMS	EduSystem	VAX
DEC/MMS	IAS	VAXcluster
DECnet	MASSBUS	VMS
DECsystem–10	PDP	VT
DECSYSTEM–20	PDT	
DECUS	RSTS	
DECwriter	RSX	digital

ZK–3307

HOW TO ORDER ADDITIONAL DOCUMENTATION
DIRECT MAIL ORDERS

USA & PUERTO RICO*

Digital Equipment Corporation
P.O. Box CS2008
Nashua, New Hampshire
03061

CANADA

Digital Equipment
of Canada Ltd.
100 Herzberg Road
Kanata, Ontario K2K 2A6
Attn: Direct Order Desk

INTERNATIONAL

Digital Equipment Corporation
PSG Business Manager
c/o Digital's local subsidiary
or approved distributor

In Continental USA and Puerto Rico call 800-258-1710.

In New Hampshire, Alaska, and Hawaii call 603-884-6660.

In Canada call 800-267-6215.

*Any prepaid order from Puerto Rico must be placed with the local Digital subsidiary (809-754-7575).

Internal orders should be placed through the Software Distribution Center (SDC), Digital Equipment Corporation, Westminster, Massachusetts 01473.

This document was prepared using an in-house documentation production system. All page composition and make-up was performed by TEX, the typesetting system developed by Donald E. Knuth at Stanford University. TEX is a trademark of the American Mathematical Society.

Contents

PREFACE ix

SECTION 1 INSTALLING THE VERSION 4.5 UPDATE KIT 1-1

1.1 THE VERSION 4.5 KIT 1-1
1.1.1 Optional Software Products 1-1
1.1.2 Requirements 1-1
1.1.3 Notes and Recommendations 1-2

1.2 APPLYING UPDATES TO VAXCLUSTER SYSTEMS 1-2
1.2.1 Updating a VAXcluster Environment: Rolling Update 1-3
1.2.2 Updating a VAXcluster Environment: Concurrent Update 1-5

1.3 PREPARING TO UPDATE YOUR SYSTEM 1-6

1.4 INSTALLING THE VERSION 4.5 UPDATE 1-10

1.5 TASKS TO PERFORM AFTER THE VERSION 4.5 UPDATE 1-14

1.6 PRINTING PATCHES APPLIED BY THE UPDATE KIT 1-17

SECTION 2 NEW AND CHANGED FEATURES 2-1

2.1 SYSTEM MANAGER INFORMATION 2-1
2.1.1 Modifying the Bootstrap Procedure to Bootstrap a VAX 8200/8300 System from an HSC-Controlled Disk 2-1
2.1.2 Alternate Nonstop Bootstrap Procedure for a VAX 8200/8300 System 2-4
2.1.3 Values to Deposit in R3 if Volume Shadowing Is Installed on a VAX 8800/8700/8550/8500 Processor 2-4
2.1.4 Defining a Remote Node for UETP Ethernet Testing 2-4
2.1.5 RESTART Option of BACKUP and Standalone BACKUP 2-5
2.1.6 New CI Port Driver Image (PADRIVER.EXE) 2-6
2.1.6.1 Supported Microcode • 2-6
2.1.6.2 Variable CI Port Sanity Timer • 2-7
2.1.6.3 Known CI Link Board Problems • 2-7

2.1.7	ADD_ Records for Numeric SYSGEN Parameters Allowed in AUTOGEN	2-7
2.2	APPLICATION PROGRAMMER INFORMATION	2-8
2.2.1	VAXBI Port Communications Controller	2-8
2.3	SYSTEM PROGRAMMER INFORMATION	2-9
2.3.1	Support for Non-DEC-Supplied Devices on the VAXBI Bus	2-9
2.3.2	Updated SYSGEN Device Table	2-9
SECTION 3	**PROBLEMS, RESTRICTIONS, AND NOTES**	3-1
3.1	GENERAL USER INFORMATION	3-1
3.1.1	Cluster Time Out of Synchronization Affects SUBMIT/AFTER Command	3-1
3.1.2	VAXTPU GET_INFO Command	3-2
3.1.3	VAXTPU and Terminal Widths	3-2
3.2	SYSTEM MANAGER INFORMATION	3-3
3.2.1	Error Count for Remote (RTA*n*:) Devices	3-3
3.2.2	SDA COPY Command Marks SYSDUMP.DMP As Empty	3-3
3.2.3	Tailored Systems and Layered Products—Installation Information	3-3
3.2.4	Permanent MONITOR Server Processes	3-5
3.2.5	Documentation Correction: X.25 Packet Level (Class 7) Events	3-5
3.2.6	Documentation Correction: Setting Up Queues for Spooled Line Printers	3-5
3.2.7	Documentation Correction: Creating a Command Procedure to Boot Standalone BACKUP from an Alternate System Root	3-6
3.2.8	Installing Optional Software Products: Use VMSINSTAL Instead of VMSUPDATE	3-6
3.2.9	Restriction on Dual-Ported Non-DSA Disks in a VAXcluster	3-6
3.2.10	Diskette Devices and the MSCP Server	3-7
3.2.11	DMB32 Layered Product Software Required for DMB32 Communications Controller	3-7
3.2.12	Protection of Security Auditing Information	3-8
3.2.13	Documentation Correction: DTE States, Substates, and State Transitions	3-9
3.3	APPLICATION PROGRAMMER INFORMATION	3-12
3.3.1	Correction to $GETLKI System Service	3-12
3.3.2	VAXBI Port Communications Controller: Error Reported While Booting Standalone BACKUP	3-13

3.3.3	SS$_NOENTRY Error Reported in XABPRO Block for ACL-Protected Files	3-13
3.3.4	Ethernet/802 Drivers: Promiscuous Mode Change Planned	3-14
3.3.5	Ethernet Controller: List of Expected Errors	3-14
3.3.6	Documentation Correction: Screen Management Routines Using AST Routines	3-15
3.3.7	SCNRTL Problems Corrected	3-15
3.3.8	TU78 Tape Driver (TFDRIVER) Error Handling	3-15
3.3.9	TE16 and TU77 Tape Driver (TMDRIVER) Corrections	3-16
3.3.10	Caution on Use of NOP Instruction as a Delay Mechanism	3-16
3.3.11	Debugging Shareable Images—Change in Behavior from Pre-Version 4.4 Releases	3-16
3.3.12	Failure of VAX BASIC SET INITIAL CHOICE Statement	3-17
3.3.13	SYS$CREMBX and Process-Private Logical Names	3-17
3.3.14	DECnet–VAX File Operations with ULTRIX-32	3-17
3.4	SYSTEM PROGRAMMER INFORMATION	3-18
3.4.1	NETACP Verification of MOP Messages	3-18
3.4.2	Documentation Correction: IFNORD, IFNOWRT, IFRD, and IFWRT Macros	3-18
3.4.3	Behavior of Zero-Length and Negative Byte Counts Submitted in $QIO Requests	3-19
3.4.4	Documentation Correction: Bootstrapping with XDELTA on VAX 8700, VAX 8550, VAX 8500, and VAX 8300 Systems	3-19
3.4.5	Documentation Correction: EXE$QIODRVPKT	3-19
3.4.6	Documentation Correction: $DEF Macro	3-20

APPENDIX A VAX/VMS VERSION 4.5 UPDATE DESCRIPTION A-1

APPENDIX B GENERIC VAXBI DEVICE SUPPORT IN VAX/VMS B-1

B.1	OVERVIEW	B-1
B.2	VAXBI CONCEPTS	B-1
B.2.1	VAXBI Address Space	B-3
B.2.2	Backplane Interconnect Interface Chip (BIIC)	B-6
B.3	INITIALIZATION PERFORMED BY VAX/VMS	B-6
B.3.1	Data Structures	B-7
B.3.2	System Control Block	B-9

B.4	INITIALIZATION PERFORMED BY THE VAXBI DEVICE DRIVER	B-10
B.4.1	Examining BIIC Self-Test Status	B-11
B.4.2	Clearing BIIC Errors, Setting Interrupts, and Enabling Interrupts	B-12
B.4.2.1	Clearing the Bus Error Register • **B-12**	
B.4.2.2	Loading the Interrupt Destination Register • **B-12**	
B.4.2.3	Setting Interrupt Vectors • **B-12**	
B.4.2.4	Enabling Error Interrupts • **B-13**	
B.4.2.5	Enabling BIIC Options • **B-13**	
B.4.3	Mapping Window Space	B-13
B.4.4	Forking from a Driver Initialization Routine	B-15
B.5	DMA TRANSFERS	B-15
B.5.1	Example: DMB32 Asynchronous/Synchronous Multiplexer	B-17
B.6	REGISTER-DUMPING ROUTINE	B-19
B.7	LOADING A VAXBI DEVICE DRIVER	B-19
B.8	REFERENCE MATERIAL	B-20
B.8.1	BIIC Register Definitions	B-21
B.8.2	IOC$ALLOSPT	B-28

INDEX

FIGURES

B–1	VAX 8200 and VAX 8300 Systems	B-2
B–2	VAX 8500, VAX 8550, VAX 8700, and VAX 8800 Systems	B-2
B–3	VAXBI Address Space	B-3
B–4	Description of VAXBI I/O Space	B-4
B–5	Physical Addresses in VAXBI I/O Space	B-5
B–6	VAXBI Device Vectors	B-10
B–7	Page Table Entry	B-17
B–8	Backplane Interconnect Interface Chip (BIIC) Registers	B-21

TABLES

1–1	Approximate Disk Block Utilization for Version 4.5 Installation Procedure	1-6
2–1	Values to Deposit in R3 in the Bootstrap Command Procedures	2-4
2–2	SYSGEN Device Table	2-10
3–1	Optional Software Products Requiring the Editing Operation for Installation on a Tailored System	3-5
3–2	(NCP–6) DTE States and Substates	3-10
3–3	(NCP–7) DTE State Transitions	3-10
B–1	Contents of the BIIC Registers	B-22

Preface

The *VAX/VMS Release Notes, Version 4.5* manual describes Version 4.5 of the VAX/VMS operating system and explains the method for updating a Version 4.4 system to Version 4.5. It lists and discusses changes to the system, new features, corrected problems, and restrictions in its use. It also describes changes and corrections to the VAX/VMS documentation set.

Intended Audience

All system users may find information of interest in this manual. Sections 2 and 3 contain notes that discuss aspects of the Version 4.5 operating system of concern to the general user, system manager, application programmer, and system programmer.

Structure of This Document

There are three major sections and two appendixes.

- Section 1 contains instructions for installing the Version 4.5 update kit.
- Section 2 briefly summarizes each new and changed system feature.
- Section 3 details Version 4.5 fixes to known problems in the operating system and published documentation. It describes restrictions that should be applied to the use of VAX/VMS Version 4.5 and contains miscellaneous technical notes as well.
- Appendix A lists the contents of the VAX/VMS update kit.
- Appendix B describes VAX/VMS support for non-DIGITAL-supplied VAXBI devices and presents some guidelines for writing a VAXBI device driver.

Associated Documents

Apart from the documents for which corrections and additions are published in Sections 2 and 3, you may find the following documents helpful while reviewing the new material presented in this manual:

- The *VAX/VMS Release Notes, Version 4.4*
- The *VAX/VMS System Manager's Reference Manual*
- The *Guide to VAX/VMS Software Installation*
- The *Guide to VAXclusters*
- The *VAX/VMS Operating System, Version 4.5 Software Product Description* (SPD 25.01.26)
- The *System Software Ordering Table* (SPD 28.98.xx)

Conventions Used in This Document

The following conventions are observed in this manual:

Convention	Meaning
RET	A symbol with a one- to six-character abbreviation indicates that you press a key on the terminal, for example, RET.
$ SHOW TIME 11-NOV-1986 11:55:22	Command examples show all output lines or prompting characters that the system prints or displays in black letters. All user-entered commands are shown in red letters.
$ TYPE MYFILE.DAT . . .	Vertical series of periods, or ellipsis, means either that not all the data that the system would display in response to the particular command is shown or that not all the data a user would enter is shown.
file-spec, . . .	Horizontal ellipsis indicates that additional parameters, values, or information can be entered.
[logical-name]	Square brackets indicate that the enclosed item is optional. (Square brackets are not, however, optional in the syntax of a directory name in a file specification or in the syntax of a substring specification in an assignment statement.)
quotation marks (") apostrophe (')	The term quotation marks is used to refer to double quotation marks ("). The term apostrophe (') is used to refer to a single quotation mark.

1 Installing the Version 4.5 Update Kit

This section outlines the procedures necessary to install the Version 4.5 update to the VAX/VMS operating system. When you install the update kit, a Version 4.5 system is produced.

1.1 The Version 4.5 Kit

The VAX/VMS Version 4.5 update kit consists of documentation, patches, and replacement files. It includes the following components:

- The *VAX/VMS Operating System, Version 4.5 Software Product Description* (SPD 25.01.26)
- The *VAX/VMS Release Notes, Version 4.5*
- Distribution media in one of the following formats:
  - Nine-track, 1600 bpi magnetic tape for all processors, including the VAX 8600, VAX 8650, and VAX 8800 processors
  - Five RX50 floppy diskettes for the VAX 8200 and VAX 8300 processors
  - Seven RX01 floppy diskettes for the VAX–11/780, VAX–11/785, or VAX–11/782 processors
  - Seven TU58 cassettes for the VAX–11/725, VAX–11/730, and VAX–11/750 processors

Appendix A lists the patches, new images, and miscellaneous fixes contained in the Version 4.5 update kit.

1.1.1 Optional Software Products

The Version 4.5 kit does not contain updates to any VAX optional software products except DECnet–VAX. For more information about optional software products, see the *System Software Ordering Table* (SPD 28.98.xx). Documentation for a specific optional software product is shipped with that product.

1.1.2 Requirements

The following cautions and restrictions *must* be observed for this update:

- The system must be running Version 4.4 prior to the application of the Version 4.5 update kit:
  - If the system being updated is not currently running VAX/VMS Version 4.4, you must upgrade it to Version 4.4 before installing the Version 4.5 update kit.
  - If you are installing VAX/VMS on a *new* system, you must install Version 4.4 before applying the Version 4.5 update.

- The Version 4.5 update kit contains several improvements to volume shadowing. If your system currently includes the volume shadowing component, you must reinstall your volume shadowing product key after the update has completed.[1] If you do not reinstall the key, you will not be able to continue to use volume shadowing.

  If you are updating a system in which the system disk is a shadow set member, see Section 4.4 of the *VAX/VMS Volume Shadowing Manual* before attempting to perform the update. Section 4.4 describes steps that you must perform before applying the update to such a disk.

- If you are installing the update on a VAX 8300 or VAX 8800 multiprocessing system, you must reinstall the multiprocessing key after completing the update. If you do not reinstall the key, you will not be running the updated images.

- If you are installing the update on a system running the VAX Ada optional software product, you must install VAX Ada Version 1.3 after applying the update.

1.1.3 Notes and Recommendations

The following notes and recommendations may help you prepare your system for the update procedure.

- If the system you are updating is a VAX 8600 or VAX 8650 system, DIGITAL recommends that you back up the console RL02 before applying the update. To do so, perform the instructions given in Section 2.8.1.1 of the *VAX/VMS System Manager's Reference Manual*, substituting "VAX 8600" for "VAX-11/780" and "RL02 disk" for "diskette" throughout the section.

1.2 Applying Updates to VAXcluster Systems

The high degree of sharing achieved among systems in a VAXcluster is the result of coordination at many levels of VAX/VMS. This level of coordination generally cannot be achieved across major or minor releases of VAX/VMS. Hence, all members of a VAXcluster must run the same version (major and minor) of VMS. In addition, VAXcluster sites must be prepared to update all VAX systems in a cluster at the same time.

An understanding of the following terms is useful in understanding the discussions in this section:

Common system root	Directory structure residing on a common system disk containing the system files that are shared by several processors in a cluster environment
Private system root	Directory structure residing in either a private, local, or shared system disk in which the system files are used by a single processor in a cluster environment
System root	Generic term referring to either a common system root or a private system root

[1] The Version 4.5 update renames all current copies of DSDRIVER.EXE in SYS$SYSTEM to DSDRIVER.V44EXE.

VAX/VMS Version 4.5 *cannot* coexist in a cluster with Version 4.3 or earlier versions of the operating system. Versions 4.5 and 4.4 may be intermixed in VAXcluster configurations, but *only* for the purpose of incrementally updating the various systems in the VAXcluster and testing the newly installed operating system on VAXcluster members.

During the time that mixed versions of VAX/VMS are operating in a cluster, you must consider the following factors:

- All systems booted from a common system root must run the same version of VAX/VMS.
- When a VAX/VMS Version 4.5 system boots in the presence of a Version 4.4 system, the system console displays the following informational message:

  ```
  %CSP-I-DIFSWVER, different versions of VAX/VMS exist in cluster
  ```

- You should complete the update from Version 4.4 to Version 4.5 on all system roots of the cluster as quickly as possible.

Given these restrictions, there are two methods of applying the update to an entire cluster:

Rolling update	Use this method for a VAXcluster that has multiple system roots (that is, any combination of private system roots and/or common system roots). Old and new versions of VAX/VMS temporarily exist simultaneously in the same cluster as you apply the update to each system root. This method thus enables old and new versions of VAX/VMS to temporarily exist together in the same VAXcluster. (See Section 1.2.1.)
Concurrent update	Use this method for a VAXcluster that has a single common system root. The entire cluster is unavailable as the update is applied to the common system root. When the update is complete, the cluster is brought back up to run the updated version. (See Section 1.2.2.)

When updating a common system root during either a rolling update or a concurrent update, you need to perform only one complete update from one of the nodes that shares the common system root. However, you may need to modify the console boot command files as well as manually invoke AUTOGEN to update the system configuration parameters. Alternatively, you may use the MAKEROOT command procedure to create new alternate roots for these nodes. (See the *Guide to VAXclusters* for additional information.)

1.2.1 Updating a VAXcluster Environment: Rolling Update

A rolling update is the method used to apply an update to a VAXcluster that has multiple system roots (that is, any combination of private and/or common system roots). In a rolling update, you apply the update to each system root individually, thus causing new and old versions of VAX/VMS temporarily to exist together in the same VAXcluster. As a result, a rolling update maintains partial system availability during an update. (See Chapter 5 of the *Guide to VAXclusters* for additional information.)

A rolling update is *not* applicable when all systems boot from a single common system root.

Perform the following steps, as appropriate, for each common system root or private system root in the cluster:

1. Check the votes and make adjustments to maintain the proper quorum that allows the cluster to continue operating throughout this process. (Chapter 5 of the *Guide to VAXclusters* describes this procedure in detail.)

2. Complete all the steps in Section 1.3 of these release notes.

   If you are updating a *private system root*, go to step 3.

   If you are updating a *common system root*, you need to perform only one complete update from one of the nodes that shares that root. For all systems on a common system root, except the one from which you will apply the update, perform the following actions:

   a. Shut down the system, using your site's standard shutdown procedure. (See Section 4.1.1 of the *VAX/VMS System Manager's Reference Manual* for a description of the SYS$SYSTEM:SHUTDOWN.COM command procedure.)

   b. After you shut down a system on a common system root, issue the following command on one of the remaining nodes:

      ```
      $ SET CLUSTER/QUORUM
      ```

      This allows one node to continue running from the common system root (assuming other nodes running from different roots supply enough votes to sustain cluster quorum).

      If proper quorum is not maintained, the shutdown procedure will hang the cluster. In this event, enter the following commands to free the cluster:

      ```
      $ CTRL/P
      >>>H
      >>>D/I 14 C
      >>>C
      IPC>Q
      IPC> CTRL/Z
      ```

3. Update the single system according to Section 1.4 of these release notes.

4. Manually reboot the updated system, as described in Section 4 of the *VAX/VMS System Manager's Reference Manual*. The updated version should now be running on the single system.

   When updating a *common system root*, reboot the other systems on the system root. This allows all systems on the common system root to run the updated version.

Note: At this point, the cluster is running with mixed versions of VAX/VMS. You should now test and verify the new version before updating the other system roots.

5. Repeat the tasks in this section, as appropriate for each system root, until all roots are running the updated version.

6. Proceed to step 2 of Section 1.5.

1.2.2 Updating a VAXcluster Environment: Concurrent Update

A concurrent update is the method used to apply an update to a VAXcluster that has a single common system root. A concurrent update is performed by shutting down the entire cluster and applying the updated version to the common system root. When the update is complete, you boot each node in the cluster to start running the updated version of VAX/VMS. All systems in the cluster are unavailable while a concurrent update is being performed.

Perform the following steps to perform a concurrent update on your VAXcluster:

1. Note the current values for all votes and quorum. You will later restore these values after the update has completed. Use the following commands:

```
$ RUN SYS$SYSTEM:SYSGEN
SYSGEN> USE CURRENT
SYSGEN> SHOW VOTES
SYSGEN> SHOW QUORUM
SYSGEN> EXIT
```

   See Chapter 5 of the *Guide to VAXclusters* for additional discussion of this procedure.

2. Shut down the entire cluster, using your site's standard shutdown procedure. (See Section 4.1.1 of the *VAX/VMS System Manager's Reference Manual* for a description of the SYS$SYSTEM:SHUTDOWN.COM command procedure.)

3. Perform a conversational boot on a single VAX system and set the votes and quorum values to 1 as follows:

```
SYSBOOT> USE CURRENT
SYSBOOT> SET VOTES 1
SYSBOOT> SET QUORUM 1
SYSBOOT> CONTINUE
```

   See Section 4.2.3 of the *VAX/VMS System Manager's Reference Manual* for further discussion of the conversational boot procedure.

4. Install the Version 4.5 update as described in Sections 1.3 and 1.4. This applies the update to the root from which the system is booted. The update procedure will automatically perform an orderly shutdown of this system when it completes.

5. Perform a conversational boot of this system, issuing the necessary SYSBOOT commands to restore the original settings of votes and quorum on this system as recorded in step 1.

6. Reboot the entire cluster according to your normal operating procedures. The entire cluster will now be running the updated version of VAX/VMS.

7. Proceed to step 2 of Section 1.5.

1.3 Preparing to Update Your System

This section describes the activities you must perform before applying the Version 4.5 update to your system. You should read this entire section before proceeding with the update.

Table 1–1 Approximate Disk Block Utilization for Version 4.5 Installation Procedure

Peak disk block utilization	11200
Net disk block utilization if files are purged during the update	6500

Perform these steps to prepare your system for the update:

1 **Back up the system disk.**

By backing up the system disk, you preserve the original system disk in the event that a system failure at a critical point in the update results in unusable or deleted files.

CAUTION: **If you elect not to back up your system disk, a system failure at a critical point of the update procedure may cause the previous contents of the disk to become irretrievable.**

To back up the system disk, proceed as follows:

a Use standalone BACKUP as described in the *Guide to VAX/VMS Software Installation* or the *VAX/VMS System Manager's Reference Manual.*

b If an additional drive with an unused disk of equal capacity is available, you can perform a disk-to-disk backup directly to it from the system disk and use the backup as the system disk during the update. To do this, you must swap the unit plugs of the two drives so that you can boot from the new backup disk using the default command procedure, DEFBOO.

If no drive with a disk of equal capacity is available, you must back up the system disk to whatever device is available:

- If the system disk is removable, remove and replace it with a spare disk. Then, transfer the files from the backup device to the spare disk by performing another backup operation. Use the spare disk as the system disk for the update and preserve the original system disk.
- If the system disk is not removable, you must use the original system disk for the update. However, you should still restore the backup from the backup device to the system disk to ensure that there is sufficient contiguous free space on the disk.

If the system you are updating is a VAX 8600 or VAX 8650 system, DIGITAL recommends that you also back up the console RL02 before applying the update. To do so, perform the instructions given in Section 2.8.1.1 of the *VAX/VMS System Manager's Reference Manual,* substituting "VAX 8600" for "VAX-11/780" and "RL02 disk" for "diskette" throughout the section.

2 Reserve space for the update files.

The VAX/VMS Version 4.5 update procedure requires that a minimum number of free blocks be available on the system disk so that the procedure can properly perform the update. To ensure that there are sufficient free blocks to meet the update procedure's *peak disk block utilization* (see Table 1–1), perform the following actions:

a Confirm the number of free blocks on the system disk by entering the following DCL command:

```
$ SHOW DEVICE SYS$SYSDEVICE:
```

b Compare the number of *free blocks* shown on the display against the required peak disk block utilization shown in Table 1–1.

If you have fewer blocks available than the peak disk block utilization figure, you must reduce the number of used disk blocks to acquire enough free space for the Version 4.5 update. DIGITAL recommends that you use the following procedure to gain the needed disk space:

a Log in to an account with sufficient privileges to create space on the system disk. DIGITAL recommends that you do not log in to the SYSTEM account. The SYSTEM account, which has all privileges (including BYPASS), is intended only for software installation, bootstrapping, and system problem diagnosis. You can avoid problems by creating another account and assigning it the minimum privileges required.

b Delete or purge all unwanted or redundant files from the system disk.

c If there still is not enough available space, copy the following files to another media and delete them from the system disk:

- All files with JNL, MAP, LOG, and STB[2] extensions
- The files SYS$ERRORLOG:ERRLOG.SYS and SYS$MANAGER:ACCOUNTNG.DAT
- All files in the directories [SYSHLP.EXAMPLES] and [SYSTEST]

If you cannot make a sufficient number of free blocks available on the system disk to meet peak utilization requirements, the update procedure will operate in an alternate mode that reduces these requirements. However, if a system failure occurs while the procedure is operating in this alternate mode, you must restore the Version 4.4 system disk from a backup copy, and restart the update procedure from the beginning.

3 Confirm the quotas and limits of the SYSTEM account.

Because you will later install the update from the SYSTEM account, you must ensure that the account has sufficient quotas and limits to successfully complete the update. To do so, perform the following actions:

a Log in to the SYSTEM account.

[2] Once the Version 4.5 update is installed, DIGITAL recommends that you copy all STB files back to their original directories, except RMS.STB (for which Version 4.5 supplies a new file).

b Run the Authorize Utility (AUTHORIZE) by entering the following commands:

```
$ SET DEFAULT SYS$SYSTEM
$ RUN AUTHORIZE
UAF> SHOW SYSTEM
```

c Compare the SYSTEM account's limits and quotas to the following values:

Open file quota (Fillm)	20
Buffered I/O limit (BIOlm)	18
Direct I/O limit (DIOlm)	18
AST limit (ASTlm)	24
Enqueue quota (Enqlm)	30
Buffered byte quota count (Bytlm)	20480

d Adjust the corresponding UAF parameters, as appropriate, to ensure that they are equal to or greater than the required values. You can change each value by entering the following command:

```
UAF> MODIFY SYSTEM/limit=new_value
```

For example:

```
UAF> MODIFY SYSTEM/DIOLM=18
```

e Return to DCL command level by issuing the following command:

```
UAF> EXIT
```

f If you have adjusted any of the SYSTEM account's values, log out and log in again so that the new values take effect.

4 Reserve sufficient global pages.

The installation procedure requires at least 50 unused global sections and 3000 unused global pages. Ensure that sufficient *unused* global sections and global pages are available to the procedure by performing the following operations:

a Display the number of *used* global sections, and *used* and *unused* global pages, by issuing the following commands:

```
$ INSTALL :== $INSTALL/COMMAND_MODE
$ INSTALL
INSTALL> LIST/GLOBAL/SUMMARY
INSTALL> EXIT
```

b Determine the *current* number of global sections by invoking the System Generation Utility (SYSGEN) and proceeding as follows:

```
$ RUN SYS$SYSTEM:SYSGEN
SYSGEN> USE CURRENT
SYSGEN> SHOW GBLSECTIONS
```

c Determine the number of *unused* global sections by subtracting the number of *used* global sections (determined in step **a** from the INSTALL display) from the current number of global sections (determined in step **b** from the SYSGEN display).

d If the number of *unused* global sections is less than 50, increase the *GBLSECTIONS* parameter,[3] using the following command:

```
SYSGEN> SET GBLSECTIONS new-value
```

To compute **new-value**, add 50 to the number of *used* global sections (determined in step **a** from the INSTALL display).

e If the number of *unused* global pages (determined in step **a** from the INSTALL display) is less than 3000, you must increase the *GBLPAGES* parameter.[3] Use the following command:

```
SYSGEN> SET GBLPAGES new-value
```

To compute **new-value**, add 3000 to the number of *used* global pages (determined in step **a** from the INSTALL display).

f Save the modified values and exit from SYSGEN using the following commands:

```
SYSGEN> WRITE CURRENT
SYSGEN> EXIT
```

g If you have modified either of the SYSGEN parameters *GBLPAGES* or *GBLSECTIONS*, use the SYS$SYSTEM:SHUTDOWN.COM command procedure to shut down the system. Reboot the system so that the new SYSGEN parameter values take effect.

5 Isolate the system from users.

Make sure that nobody but you is logged in to the system. This is a twofold procedure:

a Notify current users that they must log out.

b Ensure that no new user can log in. The following command prevents users from logging in:

```
$ SET LOGINS/INTERACTIVE=0
```

6 Shut down the network.

Perform this task only if your system is running DECnet–VAX. If you are not sure whether your system includes DECnet–VAX, enter the following command:

```
$ SHOW NETWORK
```

If the message "%SHOW-I-NONET, network unavailable" appears, skip to step 7. If your system includes DECnet–VAX, shut it down by issuing the following commands:

```
$ RUN SYS$SYSTEM:NCP
NCP> SET EXECUTOR STATE OFF
NCP> EXIT
```

[3] DIGITAL recommends that you also edit SYS$SYSTEM:MODPARAMS.DAT to reflect the modified system parameters. (Refer to the procedures described in Section 11.3 of the *VAX/VMS System Manager's Reference Manual* whenever you change SYSGEN parameters.)

7 **Stop all batch and print queues.**

To do so, perform the following tasks:

a Enter the following command to determine the state of all system queues:

```
$ SHOW QUEUE/DEVICE/BATCH/FULL/ALL
```

b Stop each active queue by issuing the command

```
$ STOP/QUEUE/NEXT queue_name
```

The NEXT qualifier allows the current job to complete before the system stops the queue. If this job may take a long time to complete, you may want to ensure that it is safe to stop it prior to completion.

8 **Review special considerations.** Under various circumstances and within certain configurations, you may be required to perform other actions before proceeding with the update. See Sections 1.1.2 and 1.1.3 to determine if any special requirements apply to your system.

1.4 Installing the Version 4.5 Update

After completing the procedures described in Section 1.3, perform the steps in this section to install the Version 4.5 update kit.

1 **Invoke the VMSINSTAL command procedure.**

Use the following command:

```
$ @SYS$UPDATE:VMSINSTAL VMS045 device-name
```

where **device-name** is the physical name of the device holding the update distribution media. Use one of the following formats for **device-name** depending on the system configuration:

- If the distribution volume (or volumes) is to be mounted on a non-HSC device, specify **device-name** using the format *ddcu*, as follows:

  dd specifies the type of device.
  c refers to the controller number.
  u refers to the device unit number.

- If the distribution volume (or volumes) is to be mounted on an HSC device, specify **device-name** using the format *hsc-name$ddcu*.

For example, if your distribution kit (load device) is a TU80 magnetic tape drive on controller A with unit number 0, you would enter the command

```
$ @SYS$UPDATE:VMSINSTAL VMS045 MUA0:
```

If you are updating from a TA80 magnetic tape drive controlled by an HSC named VICE on controller A with unit number 0, you would enter the command

```
$ @SYS$UPDATE:VMSINSTAL VMS045 VICE$MUA0:
```

If the VMSINSTAL command fails, determine whether either of the following conditions occurred:

- If VMSINSTAL displays the message "%VMSINSTAL-E-NOPRODS, None of the specified products were found", it is likely that you specified the letter "O" in the product name "VMS045" instead of a zero.

- If VMSINSTAL displays an "invalid device" error message, it will issue prompts for a device name until you specify the correct name of a device existing on the system. Remember to terminate the device name with a colon (:).

When the command succeeds, VMSINSTAL displays the following message:

```
VAX/VMS Software Product Installation Procedure V4.5

It is (date) at (time).
Enter a question mark (?) at any time for help.
```

2 Reply to VMSINSTAL prompts.

As the update procedure begins, VMSINSTAL presents its first prompt:

```
* Are you satisfied with the backup of your system disk [YES]?
```

If you are content with the current backup of the system disk, press the RETURN key and continue.

If you have not yet backed up your system disk or are otherwise dissatisfied with the current backup, perform the following operations:

a Enter *NO* and press the RETURN key. VMSINSTAL returns to DCL level to permit you to perform the backup.

b Back up and restore your system disk using standalone BACKUP as described in the *Guide to VAX/VMS Software Installation* or the *VAX/VMS System Manager's Reference Manual.* Preserve your current system disk and use the backup copy to verify that the backup copy contains a working system.

If your system is a VAX 8600 or VAX 8650 system and you have not backed up the console RL02, DIGITAL recommends that you do so now. To do so, perform the instructions given in Section 2.8.1.1 of the *VAX/VMS System Manager's Reference Manual,* substituting "VAX 8600" for "VAX-11/780" and "RL02 disk" for "diskette" throughout the section.

c Restart the update procedure at step 1 when the backup is completed.

As it proceeds, VMSINSTAL may request additional information from you or display various messages. For instance, if you did not specify the name of a load device in the command that invoked VMSINSTAL in step 1, VMSINSTAL will prompt for the name of the device holding the distribution volume:

```
* Where will the distribution volume be mounted:
```

To respond, enter the physical name of the device that will hold the distribution media during the update operation.

VMSINSTAL displays informational messages that describe the actions it is performing. During the entire process, look for error and warning messages that indicate tasks you must perform manually. Many informational messages will be displayed; these messages can usually be ignored. For instance, if you are installing from an operator's terminal, you will receive a message after each mount operation if the SYSGEN parameter *MOUNTMSG* is set and after each dismount operation if the SYSGEN parameter *DISMOUMSG* is set. Each message will appear within 30 seconds of its associated operation.

3 Mount the first (or only) volume of the update kit.

VMSINSTAL next displays the following prompt:

```
Please mount the first volume of the set on ddcu:.

* Are you ready?
```

To respond, perform the following actions:

a Insert the first (or only) distribution volume into the load device. If you are installing from diskettes, insert the first volume in the drive. If you are installing from magnetic tape, load the tape into the drive. If you are installing from TU58 cassettes, insert the first cassette into the drive.

b After you have inserted the first (or only) volume into the appropriate drive, enter *Y* and press RETURN.

VMSINSTAL then displays the following information:

```
%MOUNT-I-MOUNTED, VMS045 mounted on _ddcu:
The following products will be processed:

   VMS V4.5

        Beginning installation of VMS V4.5 at (time)
```

Note: If you are updating a tailored system or a small disk system (for instance, a system using an RK07 cartridge disk), note that when the update procedure creates free disk blocks it automatically cancels the effects of the VAX/VMS Tailoring facility. For this reason, the VMSINSTAL command procedure requires at this time that you mount the library disk so that it can save the current tailoring environment across the update procedure.

4 Select an update option.

Shortly after it has copied the first save set from the installation volume (or volumes), VMSINSTAL displays the following menu:

```
1)  Apply all fixes to the system
2)  Create a file with the descriptions of all fixes
3)  Both of the above

* What would you like to do [3]:
```

- Under option 1, VMSINSTAL performs *only* the update.
- Under option 2, VMSINSTAL does *not* perform the update. It simply creates the update description file, SYS$UPDATE:VMS045.TXT. (Appendix A lists the contents of this file.)
- Under option 3, VMSINSTAL *both* performs the update and creates the update description file.

Type one of these option numbers and press RETURN.

If you choose option 2 or 3, VMSINSTAL issues the following prompt:

```
%VMS-I-FIXDESC, The fixes are described in SYS$UPDATE:VMS045.TXT
```

5 Proceed with the update.

If you elect to proceed with the update by specifying option 1 or 3, VMSINSTAL displays the following question:

```
* Do you want to purge files replaced by this installation [YES]?
```

If you want VMSINSTAL to automatically purge files replaced by the update, press RETURN. (Refer to Section 1.5 for additional details on other ways to economize on disk space after the update has completed.) Answer *N* if you do not want these files purged. When VMSINSTAL receives your reply to this prompt, it restores the remainder of the update save sets and continues the copy operation from the specified drive.

If you are installing the update from a set of diskette or TU58 cassette volumes, VMSINSTAL automatically requests, as it completes its operations from one volume, that you remove the current volume and insert the subsequent one.

If you are installing from magnetic tape, there is only one tape for you to mount.

When VMSINSTAL completes the restoration of the save sets, it begins to apply the update to the system disk. During this time, ensure that the last (or only) volume of the update media remains mounted until the update is fully completed.

As VMSINSTAL proceeds, it displays the name of each image that is patched or installed, plus various informational messages describing the characteristics of the patches and images. You should be aware of the following situations which result in messages:

- The Patch Utility will commonly generate the following informational messages:

  ```
  %PATCH-I-NOLCL, image does not contain local symbols
  %PATCH-I-NOGBL, some or all global symbols not accessible
  ```

  These messages are a normal result of the construction of some update patches and should be ignored.

- When updating the NETACP.EXE image, VMSINSTAL displays the following:

  ```
  %PATCH-I-BRTOOFAR, destination FFFFFF74 is too far for branch operand
  ```

  This message simply informs you that a JMP instruction was used in the code generating the patch instead of a BRB or BRW instruction. The message is informational only and does not affect the validity of the patch.

- When updating an image that has already been patched, VMSINSTAL will display the following informational message:

  ```
  %PATCH-I-ECOSET, eco level nn already set in 'xxx$ROOT:filename'
  ```

  This message indicates that a patch has previously been applied, most likely during the application of the Version 4.4 mandatory update. For this reason, you can ignore messages of this sort.

  If *all* of the supplied patches in an image have already been applied, VMSINSTAL additionally displays the warning message:

  ```
  %VMSINSTAL-W-NOFILE, New file 'Filename' does not exist.
  ```

In other words, if all the necessary patches have already been made to the file, there is no need for VMSINSTAL to create a new version of the file.

At its completion, VMSINSTAL will advise you to review the various fixes in which it encountered the "NOFILE" warning message. For the previously described reasons, you can usually ignore these messages.

VMSINSTAL also creates a journal file (with a file extension of .JNL) for each image that is patched during the update process. (See Section 1.6 for additional information on the .JNL files produced by the Version 4.5 update.)

When it completes the update, VMSINSTAL displays the following message:

```
Installation of VMS V4.5 completed at (time)
```

and performs an orderly shutdown of the system.

1.5 Tasks to Perform After the Version 4.5 Update

After VMSINSTAL has completed its installation of the Version 4.5 update kit, DIGITAL recommends that you perform the following tasks:

1 **Reboot the system.** Manually reboot the system as described in Section 4 of the *VAX/VMS System Manager's Reference Manual*.

2 **Free up disk space.**

VMSINSTAL permanently uses a certain number of disk blocks (as described in Table 1–1) called the *net disk block utilization*. This figure can vary, depending on whether you chose (in step 4 of Section 1.4) to purge the old copies of system files that are replaced during the update.

Use the following methods to free up disk space:

a Confirm the free block count by issuing the command

```
$ SHOW DEVICE SYS$SYSDEVICE:
```

b Purge those files that the Version 4.5 update procedure cannot purge. In this manner you can recover approximately 2300 disk blocks. Use the PURGE command to remove old versions of the following files:

- SYS$LIBRARY:CONVSHR.EXE
- SYS$LIBRARY:ERFCTLSHR.EXE
- SYS$LIBRARY:ERFSHR.EXE
- SYS$LIBRARY:ERFSHR2.EXE
- SYS$SYSTEM:BACKUP.EXE
- SYS$SYSTEM:DCL.EXE
- SYS$SYSTEM:ERFBRIEF.EXE
- SYS$SYSTEM:ERFPROC1.EXE
- SYS$SYSTEM:F11BXQP.EXE
- SYS$SYSTEM:JOBCTL.EXE
- SYS$SYSTEM.MBXDRIVER.EXE

- SYS$SYSTEM:MTAAACP.EXE
- SYS$SYSTEM:NODRIVER.EXE
- SYS$SYSTEM:RMS.EXE
- SYS$SYSTEM:RUNDET.EXE
- SYS$SYSTEM:SYS.EXE
- SYS$SYSTEM:TTDRIVER.EXE
- SYS$UPDATE:VMSINSTAL.COM

3 Rebuild standalone BACKUP.

VAX/VMS Version 4.5 contains several corrections to images that are part of the standalone BACKUP procedure. To include these corrections, rebuild all copies of standalone BACKUP after you install Version 4.5. (This requirement includes copies on console volumes that were distributed as part of the Version 4.4 distribution kit.) Section 2.8.2.4 of the *VAX/VMS System Manager's Reference Manual* explains how to use SYS$UPDATE:STABACKIT.COM to generate a copy of standalone BACKUP.

4 Adjust system parameters.

Run the AUTOGEN procedure to adjust system parameters, issuing the command

```
$ @SYS$UPDATE:AUTOGEN SAVPARAMS SHUTDOWN
```

If, in step 4 of Section 1.3, you modified the values of *GBLSECTIONS* and *GBLPAGES* and stored their old values in SYS$SYSTEM:MODPARAMS.DAT, you may want to restore these values at this time.

For more information on the AUTOGEN procedure, see Section 11 of the *VAX/VMS System Manager's Reference Manual.*

5 Copy VMB.EXE to the console media.

When updating a VAXcluster to VAX/VMS Version 4.5, you need only apply the update once to a common system disk regardless of the number of systems that actually boot from that disk. However, it is necessary to place the Version 4.5 copy of VMB.EXE onto your system's console media.

If your system is a VAX 8800, VAX 8700, VAX 8550, or VAX 8500, you can skip this step, as the new VMB.EXE is shipped with the console media.

For a VAX–11/730 or VAX–11/725—or a VAX–11/750 that does not boot from a TU58 tape cartridge—perform the following steps:

a Invoke the System Generation Utility (SYSGEN) and connect the console by issuing the following commands:

```
$ RUN SYS$SYSTEM:SYSGEN
SYSGEN> CONNECT CONSOLE
SYSGEN> EXIT
```

b Insert the console TU58 in CSA1:.

c Copy VMB.EXE to CSA1: using the Exchange Utility, as follows:

```
$ EXCHANGE
EXCHANGE> COPY/LOG SYS$SYSTEM:VMB.EXE CSA1:
EXCHANGE> EXIT
```

For all other VAX processors, invoke the console update command procedure SYS$UPDATE:UPDATE_CONSOLE.COM as follows:

```
$ @SYS$UPDATE:UPDATE_CONSOLE
```

If your system is a VAX 8650, VAX 8600, VAX 8300, or VAX 8200, this procedure will simply copy the new file onto your existing console media.

If your system is a VAX-11/780, VAX-11/782, VAX-11/785, or VAX-11/750, this procedure will use EXCHANGE to save the contents of your existing console. It will then merge the new files on the saved copy of your console media. Finally, it will request that you insert a scratch medium so that it can create new console media containing the new file. Your original console media will not be modified.

6 Modify MEMSIZE parameter.

If you have applied the Version 4.5 update to a VAX 8300 configuration that includes 32MB of memory, you must modify the configuration parameter MEMSIZE so that it reflects the correct size of physical memory in pages. For 32MB systems, this parameter should be set to 65536.

On such a VAX 8300 configuration, the SHOW MEMORY command will incorrectly display the amount of physical memory present on the system. An effect of this problem is that unless you modify the MEMSIZE parameter, the VAX/VMS AUTOGEN command procedure will incorrectly set up several system parameters. This problem will be corrected in a future release of VAX/VMS.

To modify the MEMSIZE parameter, follow these steps:

a Edit the file SYS$SYSTEM:MODPARAMS.DAT by adding the following DCL assignment statement:

```
MEMSIZE = 65536
```

b Verify this modification to MODPARAMS.DAT by issuing the following command:

```
$ @SYS$UPDATE:AUTOGEN GETDATA
```

c Examine the resulting PARAMS.DAT file.

d Once you have verified the PARAMS.DAT file, issue the following command:

```
$ @SYS$UPDATE:AUTOGEN GENPARAMS SETPARAMS
```

e Reboot your system by issuing the following command:

```
$ @SYS$UPDATE:AUTOGEN REBOOT
```

f When the AUTOGEN REBOOT operation completes, verify the changes in AUTOGEN.PAR.

For additional discussion of the above procedure, refer to Section 11.4 of the *VAX/VMS System Manager's Reference Manual*.

1.6 Printing Patches Applied by the Update Kit

If you select either option 2 or 3 as an update option (in step 4 of Section 1.4), VMSINSTAL produces the update description file, SYS$UPDATE:VMS045.TXT. This file lists the patches, new images, and miscellaneous fixes that are part of the Version 4.5 update kit. If you print this file, you will obtain the listing that appears in Appendix A of these release notes.

If you select either option 1 or 3, VMSINSTAL produces a journal file (with the extension .JNL) for each image that is patched during the update. Journal files contain a record of each patch made to these images but do not contain information about modules that are replaced. Note that no journal files are created for tailored systems.

If you want a listing of the patches produced by the update process, print the journal files using the following steps:

1. Complete the update procedure that installs Version 4.5, including rebooting the system as described in the software installation guide for your processor.
2. Log in to any account that has SYSPRV privilege and issue the following command:

```
$ PRINT SYS$SYSTEM:*.JNL,SYS$LIBRARY:*.JNL
```

The journal files produced by the Version 4.5 update procedure occupy approximately 700 blocks. If you must conserve disk space, you may want to delete these files from the system disk after you print them.

2 New and Changed Features

This section discusses new features added to the VAX/VMS operating system in Version 4.5. It also describes features that have changed since the release of Version 4.4.

For ease of reference, the material in this section is arranged under the following categories:

Section 2.1—System Manager Information
Section 2.2—Application Programmer Information
Section 2.3—System Programmer Information

To find specific topics, consult the index in the back of this manual.

2.1 System Manager Information

The following section describes the new features of VAX/VMS Version 4.5 of interest to the system manager. It also discusses changes to the operating system since Version 4.4.

2.1.1 Modifying the Bootstrap Procedure to Bootstrap a VAX 8200/8300 System from an HSC-Controlled Disk

This note provides information that was omitted from Sections 2.4 and 2.5 of the *VAX/VMS System Manager's Reference Manual*.

If you will bootstrap your 8200/8300 processor from a local (non-HSC) disk, no default bootstrap command procedure is required. If you will bootstrap from an HSC-controlled disk, modify the CIBOO.CMD command procedure that is supplied on the console diskette and rename it DEFBOO.CMD. When the console diskette is in the console diskette drive and contains a file named DEFBOO.CMD, the processor uses DEFBOO.CMD to perform either of the following actions:[1]

1 Reboot automatically

2 Bootstrap the processor when you enter the BOOT command at the console mode prompt without specifying a device name

Follow this procedure to modify the CIBOO.CMD command procedure and rename it DEFBOO.CMD.

1 Be sure your console device is connected. If it is not, invoke SYSGEN and issue the following command to connect it:

```
SYSGEN> CONNECT CONSOLE
SYSGEN> EXIT
```

[1] These actions occur assuming that the default boot descriptors in the processor's EEPROM are set to boot your system. An automatic reboot occurs only if the lower key switch on the system control panel is set to "Autostart."

2 Insert the console diskette into the console drive (CSA1:, the left-hand diskette drive).

3 Enter the following command to mount the console diskette:

```
$ MOUNT/FOREIGN CSA1:
```

4 Use the DXCOPY.COM command procedure to copy CIBOO.CMD to a disk directory. You cannot modify a file directly on the console diskette because of the way the diskette is formatted. Copy CIBOO.CMD to a disk directory as follows:

a Enter the following command to invoke DXCOPY.COM command procedure:

```
$ @SYS$UPDATE:DXCOPY
```

b Enter Y in response to the following prompt:

```
Is the system console storage medium mounted (Y/N)?: Y
```

c Enter Y in response to the following prompt:

```
Copy from console medium (Y/N)?: Y
```

d Enter CIBOO.CMD in response to the following prompt:

```
Name of file to be copied?: CIBOO.CMD
$
```

e DXCOPY copies the file to your default directory and exits to DCL command level.

5 When the DCL prompt appears, use a text editor to edit CIBOO.CMD. Originally, this file contains the following text:

```
!CIBOO.CMD   :Boot command file to boot a VAX 8200/8300 from an HSC disk.
!
!
!  Note "n", "p" (and "q"), "u", and "r" are single hexadecimal characters
!
D/G 0 20               ! CI Port Device Type Code
!D/G 1 n               ! n = CI adapter's VAXBI node number
!D/G 2 p               ! Use the HSC controller at CI node p
!D/G 2 0p0q            ! Use either the HSC controller at CI nodes p and q
!D/G 3 u               ! u = Disk drive unit number
D/G 4 0                ! Boot Block LBN (not used)
!D/G 5 r0000000        ! r = system root [SYSR...], Software boot flags
D/G E 200              ! Address of Working Memory+^X200
LOAD VMB.EXE/START:200 ! Load Primary Bootstrap
START 200              ! Start Primary Bootstrap
```

Edit CIBOO.CMD as follows. All numbers you insert in this file are in hexadecimal radix.

a Delete the comment character (!) that appears before the D/G 1 command and replace **n** with the VAXBI node number of the CI adapter.

b If your processor is connected to one HSC controller, delete the comment character (!) that appears before the first D/G 2 command and replace **p** with the HSC controller number. If your processor is connected to two HSC controllers, delete the comment character (!) that appears before the second D/G 2 command and replace **p** with the VAXBI node number of the first HSC and replace **q** with the

VAXBI node number of the second HSC. Note that you can delete the comment character from only one of these commands.

c Delete the comment character (!) that appears before the D/G 3 command and replace **u** with the unit number of the HSC disk from which you will bootstrap the VAX/VMS operating system.

d Delete the comment character (!) that appears before the D/G 5 command and replace **r** with the number of the system root from which you will bootstrap the VAX/VMS operating system. By default, the VAX/VMS operating system is stored in system root 0.

e Exit from the text editor.

6 Enter the following command to rename CIBOO.CMD to DEFBOO.CMD.

```
$ RENAME CIBOO.CMD DEFBOO.CMD
```

7 Use the DXCOPY.COM command procedure to copy DEFBOO.CMD to the console diskette.

a Enter the following command to invoke the DXCOPY.COM command procedure:

```
$ @SYS$UPDATE:DXCOPY
```

b Enter Y in response to the following prompt:

```
Is the system console storage medium mounted (Y/N)?: Y
```

c Enter N in response to the following prompt:

```
Copy from console medium (Y/N)?: N
```

The negative response tells DXCOPY you want to copy the file from your default directory to the console storage medium.

d Enter DEFBOO.CMD in response to the following prompt:

```
Name of file to be copied?: DEFBOO.CMD
$
```

DXCOPY copies DEFBOO.CMD to the console volume and exits to DCL command level.

8 Dismount and remount the console diskette using the following commands.

```
$ DISMOUNT CSA1:
$ MOUNT CSA1:
```

You have successfully created a default bootstrap command procedure, DEFBOO.CMD, on the console diskette.

2.1.2 Alternate Nonstop Bootstrap Procedure for a VAX 8200/8300 System

The following text adds information to that given in Section 4.2.2 of the *VAX/VMS System Manager's Reference Manual*:

You normally use an alternate nonstop bootstrap command procedure when the default bootstrap procedure cannot be accessed because of problems with the device designated in the default procedure. To bootstrap the system using an alternate nonstop procedure, follow these steps:

1 Halt the processor. You halt a VAX 8200 or VAX 8300 system by pressing CTRL/P.

2 Bootstrap the system, using the following command:

For the VAX 8200 and VAX 8300

```
>>> B ddnu
```

The code *dd* is the device type, *n* is the VAXBI node number, and *u* is the unit number for the disk on which the alternate bootstrap procedure resides.

2.1.3 Values to Deposit in R3 if Volume Shadowing Is Installed on a VAX 8800/8700/8550/8500 Processor

Section 3.3.1 of the *VAX 8800/8700/8550/8500 Operations Guide* describes the procedure for modifying the template bootstrap command procedures BCI*aaa*.COM, BDA*aaa*.COM, and UDA*aaa*.COM (where *aaa* is BOO, GEN, or XDT) to bootstrap your processor. If volume shadowing is installed on your processor, modify the bootstrap command procedures to deposit the values listed in Table 2–1 in R3.

Modify registers R0 through R2 and R4 through R5 to contain the values listed in Table 3–2 of the *VAX 8800/8700/8550/8500 Operations Guide*. See the *VAX/VMS Volume Shadowing Manual* for more information about volume shadowing.

Table 2–1 Values to Deposit in R3 in the Bootstrap Command Procedures

Bit Position	Possible Values	Meaning
<31:24>	80_{16}	Shadow set indicator
<23:16>	*uu*	Shadow unit number (DUS*uu*)
<15:00>	*uu*	Unit number of booting member of shadow set

2.1.4 Defining a Remote Node for UETP Ethernet Testing

When the UETUNAS00 test of the User Environment Test Package (UETP) executes, it is sometimes difficult to determine whether the problems it reports concern the device under test or the remote device. The easiest way to ensure that the test properly reports errors on the device under test is to define a "good turnaround". A "good turnaround" is a remote node that you know turns around Ethernet packets correctly and is up and waiting in the ready state.

You can make the UETUNAS00 test use a known "good turnaround" by performing the following actions. In the commands that follow, assume that the "good" device is on node BETA, and that node BETA is already defined in the network database.

1 Find the address of the "good" Ethernet node by using the Network Control Program (NCP). In order to use NCP, the following conditions must apply:

- DECnet must be up and running on the system.
- The account you are using must have TMPMBX and NETMBX privileges.

Use the following commands:

```
$ RUN SYS$SYSTEM:NCP
NCP> TELL BETA SHO CHAR ACTIVE LINES
```

If node BETA has not been defined in your network database, NCP displays an error message. In this event, specify another "good" node and retry the command. Otherwise, see your system or network manager.

NCP displays information similar to the following:

```
Active Line Volatile Characteristics as of  15-OCT-1986 16:13:02

                Line = UNA-0

                Counter timer            = 28800
                Receive buffers          = 6
                Controller               = normal
                Protocol                 = Ethernet
                Service timer            = 4000
                Hardware address         = AA-00-04-00-46-D3
                UNA device buffer size   = 1498
```

2 Use the displayed *hardware address*—in this case, AA00040046D3—to define the logical name TESTNIADR to point to the "good turnaround." Note that you do *not* specify the hyphens (-).

First, log in to the SYSTEST account. Then enter the following command:

```
$ DEFINE/SYSTEM TESTNIADR AA00040046D3
```

3 Run the UETP.

4 When UETP has completed, deassign the logical name TESTNIADR by issuing the following command:

```
$ DEASSIGN/SYSTEM TESTNIADR
```

2.1.5 RESTART Option of BACKUP and Standalone BACKUP

If, in the course of writing a save set to tape, BACKUP or the Standalone Backup Utility encounters bad media, or other excessive hardware or media-related errors, the utility will generate the following informational message:

```
%BACKUP-I-SPECIFY, specify option ('valid options')
```

(See page 2–523 of the *VAX/VMS System Messages and Recovery Procedures Reference Manual* for a complete description of this message.)

If the output volume is the first volume in the backup operation, only QUIT and CONTINUE are available as valid recovery options. If the output volume is some subsequent volume in the backup operation, then RESTART is also available.

RESTART causes BACKUP or standalone BACKUP to restart the backup operation at the beginning of the current save set volume. As of Version 4.5, the utility unloads the current tape from the drive as soon as the RESTART option is taken and then prompts for a replacement volume. It is important that the operator *not* load the new tape until the utility has prompted for it.

Prior to Version 4.5, if more than a threshold number of errors were detected on the output tape and the operator wanted to select the RESTART option, the operator had to remove the tape from the drive before replying to the BACKUP prompt. On tape drives attached to an HSC this caused the tape drive to disappear from the controller's table of valid devices. In this event, the entire backup operation had to be restarted.

2.1.6 New CI Port Driver Image (PADRIVER.EXE)

VAX/VMS Version 4.5 contains a new image of the CI port driver, PADRIVER.EXE.

2.1.6.1 Supported Microcode

All sites should upgrade to Version 7.0 of the CI–780 microcode as soon as possible. This microcode fixes the following problems that occur in large clusters:

- Miscellaneous Error #5, Internal Queue Retry Expired
- Arbitration Timeout
- Buffer Length Violation

If you see errors of these kinds in the error log file, have field service upgrade the CI microcode to Version 7.0 as quickly as possible.

You can identify the current microcode version by following these steps:

1 Execute the following DCL command:

```
$ SHOW CLUSTER/CONTINUOUS
```

2 Enter the ADD RP_REVIS subcommand:

```
COMMAND> ADD RP_REVIS
```

The low-order word is the RAM version and the high-order word is the PROM version. For Version 7.0 microcode, this field will contain 70007_{16}.

The port driver will display the following message for sites containing old versions of the microcode:

```
%PAA0, - CI port ucode not at current rev level.  PROM/RAM rev is 0005/0003
```

2.1.6.2 Variable CI Port Sanity Timer

Version 7.0 of the CI microcode contains a variable sanity timer. When this sanity timer expires, the following error message will appear on the operator's console. The message will show that the Port Status Register (PSR) has a value of 40_{16}.

```
%PAA0, - Port Error Bit(s) Set - CNF/PMC/PSR xxxxxxxx/xxxxxxxx/00000040
```

The appearance of this error and other CI-related timeouts does not necessarily mean that the CI hardware is bad. The system could be spending a long time at high hardware priority levels. This long latency could result from the setting of the SYSGEN parameters, the nature of the processing load on the cluster, or the presence of user-written privileged code.

You should first increase the *PASTIMOUT* parameter until these errors occur infrequently, if at all. You may then wish to consult the *Guide to VAX/VMS Performance Management* to investigate the general performance characteristics of this system. For more information on CI-related timeouts, consult the *VAX/VMS Release Notes, Version 4.3* and the *VAX/VMS Release Notes, Version 4.4*.

2.1.6.3 Known CI Link Board Problems

The CI Link Board handles transmission and reception of packets on the CI wire. This board has known problems which can cause a high rate of the following "PA" device entries in the system error log file:

```
PATH #n WENT FROM GOOD TO BAD
PATH #n WENT FROM BAD TO GOOD
CABLES HAVE GONE FROM UNCROSSED TO CROSSED
CABLES HAVE GONE FROM CROSSED TO UNCROSSED
PORT HAS CLOSED VIRTUAL CIRCUIT
SOFTWARE IS CLOSING VIRTUAL CIRCUIT
```

These errors generally appear in large clusters after the addition of a new machine to the cluster. These errors are usually not fatal to the operation of the cluster. You should first take precautions to keep the error log file from filling the system disk. You should then check with DIGITAL field service for recommendations on reducing the rate of these soft errors.

2.1.7 ADD_ Records for Numeric SYSGEN Parameters Allowed in AUTOGEN

With Version 4.5, AUTOGEN allows ADD_ records to be included in SYS$SYSTEM:MODPARAMS.DAT for all numeric SYSGEN parameters. Previous to this release, an ADD_ record would affect only those parameters that AUTOGEN itself calculated, the amount specified by the record being added to AUTOGEN's calculated value. (See Section 11.4 of the *VAX/VMS System Manager's Reference Manual.*)

In Version 4.5, the value specified in an ADD_ record for a parameter that AUTOGEN does not calculate will be added to that parameter's default value. For example, if AUTOGEN encounters the record "ADD_WSINC=50" in MODPARAMS.DAT, *WSINC* will be set to *200* (the default of 150 plus the specified 50) after the next boot.

2.2 Application Programmer Information

The following section describes the new features of VAX/VMS Version 4.5 of interest to the application programmer. It also discusses changes to the operating system since Version 4.4.

2.2.1 VAXBI Port Communications Controller

VAX/VMS Version 4.5 supports a new VAXBI port communications controller, connecting to the VAXBI bus.

To configure a VAXBI port communications controller on a system that also incorporates a KLESI–B or DWBUA module (the UNIBUS of which includes a TU81 or TU81–PLUS magnetic tape subsystem), you must adhere to the following restrictions:

- If the KLESI–B and/or DWBUA modules are on the same VAXBI bus as the VAXBI port communications controller, you must place the VAXBI port communications controller at a higher node ID than the KLESI–B and DWBUA.
- If the KLESI–B and/or DWBUA modules are on different VAXBI buses (as can be the situation on a VAX 8500/8550/8700/8800 system), you must place the VAXBI port communications controller module on a higher-numbered VAXBI bus than the KLESI–B and DWBUA.

These restrictions will be removed in a future release of VAX/VMS.

The QIO interface to the VAXBI port communications controller is the same as that described for the DEUNA, DEQNA, and DELUA device drivers in Section 6 of the *VAX/VMS I/O User's Reference Manual: Part II*, with the following exceptions:

- The device type of the VAXBI port communications controller is DT$_ET_DEBNT.
- The packet size restrictions that apply to the DELUA when the controller is in loopback mode also apply to the VAXBI port communications controller.
- The NMA$C_PCLI_CRC parameter is applicable to the VAXBI port communications controller.
- The NMA$C_PCLI_ILP parameter is applicable to the VAXBI port communications controller.
- The 802 QIO interface is not supported in Version 4.5. The 802 QIO interface will be supported in a future release of VAX/VMS.

The VAXBI port communications controller is supported by ETDRIVER. Its device name is ET*cu* where *c* is the controller and *u* is the unit number (for example, ETA0).

The NCP LINE and CIRCUIT name for the VAXBI port communications controller is BNT-*controller-number* (for example, BNT-0 for ETA*n*, BNT-1 for ETB*n*).

To use the VAXBI port communications controller with the LAT terminal server, add the following command to your system startup command procedure:

```
$ DEFINE/SYSTEM LAT$DEVICE ETA0
```

2.3 System Programmer Information

The following section describes the new features of VAX/VMS Version 4.5 of interest to the system programmer. It also discusses changes to the operating system since Version 4.4.

2.3.1 Support for Non-DEC-Supplied Devices on the VAXBI Bus

VAX/VMS Version 4.5 provides certain generic support in the system initialization routines of the VAX 8200, VAX 8300, VAX 8500, VAX 8550, VAX 8700 and VAX 8800 processors for customer devices attached to a VAXBI bus. A description of this support, including suggestions for customer-written device drivers, appears in Appendix B of this manual.

2.3.2 Updated SYSGEN Device Table

The following text and table update the description of the SYSGEN device table, published in both the Version 4.4 *Writing a Device Driver for VAX/VMS* manual and the Version 4.4 *VAX/VMS System Generation Utility Reference Manual*.

The SYSGEN device table (see Table 2–2) lists the characteristics of all DIGITAL devices. This table indicates the following information for each device type:

- Device name
- Device controller name
- Interrupt vector
- Number of interrupt vectors per controller
- Vector alignment factor
- Address of the first device register for each controller recognized by SYSGEN (the first register is usually, but not always, the control and status register (CSR))
- Number of registers per controller
- Device driver name
- Indication of whether the driver is or is not supported

Devices not listed in the SYSGEN device table include the following:

- Non-DIGITAL-supplied devices with fixed CSR and vector addresses. These devices have no effect on autoconfiguration. Customer-built devices should be assigned CSR and vector addresses beyond the floating address space reserved for DIGITAL-supplied devices.
- Those DIGITAL-supplied, floating-vector devices that the AUTOCONFIGURE command does not recognize. Use the CONNECT command to attach these devices to the system.

Table 2–2 SYSGEN Device Table

Device Name	Controller Name	Vector	Number of Vectors	Alignment	CSR/Rank	Number of Registers	Driver Name	Support
CR	CR11	230	1	—	777160	—	CRDRIVER	Yes
DM	RK611	210	1	—	777440	—	DMDRIVER	Yes
LP	LP11	200 170 174 270 274	—	—	777514 764004 764014 764024 764034	—	LPDRIVER	Yes
DL	RL11	160	1	—	774400	—	DLDRIVER	Yes
MS	TS11	224	1	—	772520	—	TSDRIVER	Yes
DY	RX211	264	1	—	777170	—	DYDRIVER	Yes
DQ	RB730	250	1	—	775606	—	DQDRIVER	Yes
PU	UDA	154	1	—	772150	—	PUDRIVER	Yes
PT	TU81	260	1	—	774500	—	PUDRIVER	Yes
XE	UNA	120	1	—	774510	—	XEDRIVER	Yes
XQ	QNA	120	1	—	774440	—	XQDRIVER	Yes
OM	DC11	Float	2	8	774000 774010 774020 774030 . . . 32 units maximum	—	OMDRIVER	No
DD	TU58	Float	2	8	776500 776510 776520 776530 . . . 16 units maximum	—	DDRIVER	Yes
OB	DN11	Float	1	4	775200 775210 775220 775230 . . . 16 units maximum	—	OBDRIVER	No

Table 2–2 (Cont.) SYSGEN Device Table

Device Name	Controller Name	Vector	Number of Vectors	Alignment	CSR/Rank	Number of Registers	Driver Name	Support
YM	DM11B	Float	1	4	770500 770510 770520 770530 . . . 16 units maximum	—	YMDRIVER	No
OA	DR11C	Float	2	8	767600 767570 767560 767550 . . . 16 units maximum	—	OADRIVER	No
PR	PR611	Float	1	8	772600 772604 772610 772614 . . . 8 units maximum	—	PRDRIVER	No
PP	PP611	Float	1	8	772700 772704 772710 772714 . . . 8 units maximum	—	PPDRIVER	No
OC	DT11	Float	2	8	777420 777422 777424 777426 . . . 8 units maximum	—	OCDRIVER	No
OD	DX11	Float	2	8	776200 776240	—	ODDRIVER	No

Table 2–2 (Cont.) SYSGEN Device Table

Device Name	Controller Name	Vector	Number of Vectors	Alignment	CSR/Rank	Number of Registers	Driver Name	Support
YL	DL11C	Float	2	8	775610 775620 775630 775640 . . . 31 units maximum	—	YLDRIVER	No
YJ	DJ11	Float	2	8	Float	4	YJDRIVER	No
YH	DH11	Float	2	8	Float	8	YHDRIVER	No
OE	GT40	Float	4	8	772000 772010	—	OEDRIVER	No
LS	LPS11	Float	6	8	770400	—	LSDRIVER	No
OR	DQ11	Float	2	8	Float	4	ORDRIVER	No
OF	KW11W	Float	2	8	772400	—	OFDRIVER	No
XU	DU11	Float	2	8	Float	4	XUDRIVER	No
XW	DUP11	Float	2	8	Float	4	OODRIVER	No
XV	DV11	Float	3	8	775000 775040 775100 775140	—	XVDRIVER	No
OG	LK11	Float	2	8	Float	4	OGDRIVER	No
XM	DMC11	Float	2	8	Float	4	XMDRIVER	Yes
TTA	DZ11	Float	2	8	Float	4	DZDRIVER	Yes
XK	KMC11	Float	2	8	Float	4	XKDRIVER	No
OH	LPP11	Float	2	8	Float	4	OHDRIVER	No
OI	VMV21	Float	2	8	Float	4	OIDRIVER	No
OJ	VMV31	Float	2	8	Float	8	OJDRIVER	No
OK	DWR70	Float	2	8	Float	4	OKDRIVER	No
DL	RL11	Float	1	4	Float	4	DLDRIVER	Yes
MS	TS11	Float	1	4	772524 772530 772534	—	TSDRIVER	Yes
LA	LPA11	Float	2	8	770460	—	LADRIVER	Yes
LA	LPA11	Float	2	8	Float	8	LADRIVER	Yes
OL	KW11C	Float	2	8	Float	4	OLDRIVER	No
RSV	RSV	Float	1	8	Float	4	RSVDRIVER	No
DY	RX211	Float	1	4	Float	4	DYDRIVER	Yes
XA	DR11W	Float	1	4	Float	4	XADRIVER	Yes
XB	DR11B	124	—	—	772410	—	XBDRIVER	No

Table 2–2 (Cont.) SYSGEN Device Table

Device Name	Controller Name	Vector	Number of Vectors	Alignment	CSR/Rank	Number of Registers	Driver Name	Support
XB	DR11B	Float	1	4	772430	—	XBDRIVER	No
XB	DR11B	Float	1	4	Float	4	XBDRIVER	No
XD	DMP11	Float	2	8	Float	4	XDDRIVER	Yes
ON	DPV11	Float	2	8	Float	4	ONDRIVER	No
IS	ISB11	Float	2	8	Float	4	ISDRIVER	No
XD	DMV11	Float	2	8	Float	8	XDDRIVER	No
XE	UNA	Float	1	4	Float	4	XEDRIVER	No
XQ	QNA	Float	1	4	774460	—	XQDRIVER	Yes
PU	UDA	Float	1	4	Float	2	PUDRIVER	Yes
XS	KMS11	Float	3	8	Float	8	XSDRIVER	No
XP	PCL11	Float	2	8	764200 764240 764300 764340	—	XPDRIVER	No
VB	VS100	Float	1	4	Float	8	VBDRIVER	No
PT	TU81	Float	1	4	Float	2	PUDRIVER	Yes
OQ	KMV11	Float	2	8	Float	8	OQDRIVER	No
UK	KCT32	Float	2	8	764400 764440 764500 764540	—	UKDRIVER	No
IX	IEQ11	Float	2	8	764100	—	IXDRIVER	No
TX	DHV11	Float	2	8	Float	8	YFDRIVER	Yes
DT	TC11	214	1	—	777340	—	DTDRIVER	No
VC	VCB01	Float	2	1	777200	—	VCDRIVER	Yes
OT	LNV11	Float	1	4	776200	—	OTDRIVER	No
ZQ	QTA	Float	1	4	772570	—	ZQDRIVER	No
ZQ	QTA	Float	1	4	Float	4	ZQDRIVER	No
OP	DSV11	Float	1	4	Float	4	OPDRIVER	No
OU	ADV11C	Float	2	4	770400	—	OUDRIVER	No
OU	ADV11C	Float	2	4	Float	1	OUDRIVER	No
OV	AAV11C	Float	0	4	770440	—	OVDRIVER	No
OV	AAV11C	Float	0	4	Float	2	OVDRIVER	No
AX	AXV11C	140	2	—	776400	—	AXDRIVER	No
AX	AXV11C	Float	2	4	Float	2	AXDRIVER	No
KZ	KWV11C	Float	2	4	770420	—	KZDRIVER	No
KZ	KWV11C	Float	2	4	Float	1	KZDRIVER	No

Table 2–2 (Cont.) SYSGEN Device Table

Device Name	Controller Name	Vector	Number of Vectors	Alignment	CSR/Rank	Number of Registers	Driver Name	Support
AZ	ADV11D	Float	2	4	776410	—	AZDRIVER	No
AZ	ADV11D	Float	2	4	Float	2	AZDRIVER	No
AY	AAV11D	Float	2	4	776420	—	AYDRIVER	No
AY	AAV11D	Float	2	4	Float	2	AYDRIVER	No
VA	VCB02	Float	3	16	777400 777402 777404 777406 . . . 8 units maximum	—	VADRIVER	Yes

3 Problems, Restrictions, and Notes

This section discusses problems that have been corrected in Version 4.5 of the VAX/VMS operating system. It also describes any restrictions that may apply to the use of the Version 4.5 operating system, and contains other information concerning the release.

For ease of reference, the material in this section is arranged under the following categories:

Section 3.1—General User Information
Section 3.2—System Manager Information
Section 3.3—Application Programmer Information
Section 3.4—System Programmer Information

To find specific topics, consult the index in the back of this manual.

3.1 General User Information

This section describes problems resolved in VAX/VMS Version 4.5, lists known restrictions, and contains other information about the release of interest to the general user.

3.1.1 Cluster Time Out of Synchronization Affects SUBMIT/AFTER Command

In a VAXcluster, a batch job submitted to execute at a specified time may begin execution a little before or after the requested time. This occurs when the clocks of the member systems in the VAXcluster are not synchronized. For example, a job submitted using the DCL command SUBMIT/AFTER=TOMORROW may execute at 23:58 relative to the host system's clock.

This problem can occur in a cluster even if a job is run on the same machine from which it was submitted, because the redundancy built into the batch/print system allows more than one job controller in the cluster to receive a timer AST for the job and, thus, to schedule it for execution. Moreover, this behavior is exacerbated if the batch job immediately resubmits itself to run the next day using the same SUBMIT command. This can result in having multiple instances of the job executing simultaneously because TOMORROW (after midnight) may be only a minute or two in the future.

A workaround to this problem is to place the SUBMIT command in a command procedure that begins with a WAIT command, where the *delta-time* specified in the WAIT command is greater than the maximum difference in time between any two systems in the cluster. Use the SHOW TIME command on each system to determine this difference in time.

3.1.2 VAXTPU GET_INFO Command

Version 4.5 corrects a VAXTPU problem in which the GET_INFO built-in procedure would cause an access violation when too few arguments were passed to it. GET_INFO code has been fixed to check the number of parameters properly.

3.1.3 VAXTPU and Terminal Widths

Version 4.5 corrects a problem that occurred when VAXTPU set up a terminal width other than 80, 132, or 84. Until this release, VAXTPU forced to 80 a terminal width less than or equal to 80. VAXTPU forced to 132 a terminal width greater than 80, and treated a device type of VK100 as a special case of a terminal width of 84. This behavior is undesirable on such systems as the VAXstation, which permits terminal widths other than 80 and 132.

The Version 4.5 replacement image TPU$CCTSHR.EXE corrects this problem for terminal widths other than 80 or 132 by preserving the terminal width at startup. If a terminal width *other than* 80 or 132 is specified, in either a SET WIDTH command issued through the EVE interface or a SET SCREEN command issued through the EDT emulator interface, VAXTPU will *not* send an escape sequence to the terminal to change the width to 80 or 132. (This caused the character size to change on a VT100 or VT200, and the font to change on a VAXstation.)

For a terminal width of 80 or 132, there is no change in behavior.

If, for some applications, it is important to preserve the old behavior, you can access both the EVE interface and the EDT emulator interface sources in directory SYS$LIBRARY. Change the interface accordingly:

- If the terminal width to be set is *less than* 80, first set the width to 80. Then set it to the desired width.
- If the terminal width to be set is *greater than* 80, first set the width to 132. Then set it to the desired width.

Remember to record any changes you make to the EVE interface or EDT emulator interface sources. Future releases of the operating system may supply new versions of the sources and require that you redo your edits.

The following documentation corrections correspond to these changes:

Please correct the description of the SET (WIDTH, . . .) built-in procedure on page 4–208 of the *VAX Text Processing Utility Reference Manual* by replacing the first two paragraphs with the following text:

> If a SET (WIDTH, . . .) command causes a window to become wider than the current screen width, VAXTPU changes the screen width to the specified width.
>
> If a SET (WIDTH, . . .) command causes a window to be less than the screen width, VAXTPU changes the screen width to the specified width if all the other visible
> the screen width.

On page F–15 of the same manual, please delete the second sentence in the description of the *n* parameter of the EVE interface's SET WIDTH command.

3.2 System Manager Information

This section describes problems resolved in VAX/VMS Version 4.5, lists known restrictions, and contains other information about the release of interest to the system manager.

3.2.1 Error Count for Remote (RTA*n*:) Devices

The error count for remote terminals may be randomly incremented due to a software protocol error. You can observe the error count via the DCL command SHOW DEVICE RTA*n*:. The increase in error count, however, does not reflect a hardware error or any other data corruption.

This problem will be fixed in a future release of the operating system.

3.2.2 SDA COPY Command Marks SYSDUMP.DMP As Empty

The correct behavior of the System Dump Analyzer (SDA) COPY command varies depending on whether the crash dump is in PAGEFILE.SYS or SYSDUMP.DMP. In the former case, page file pages should be released (that is, the copy of the dump in PAGEFILE.SYS is lost) when the copy completes. In the latter, the original dump in SYSDUMP.DMP should be retained until the next time the system crashes or is shut down.

In Version 4.5, SDA incorrectly marks SYSDUMP.DMP as empty after a successful copy, indicating that the data in SYSDUMP.DMP is no longer accessible to SDA. The dump should be analyzed using the dump file created by the SDA COPY command.

This problem will be corrected in a future release.

3.2.3 Tailored Systems and Layered Products—Installation Information

This note corrects information appearing in Section 3.2.27 of the *VAX/VMS Release Notes, Version 4.4*.

Sites utilizing supported small-disk tailored systems must perform the following editing operation before installing any of the VAX/VMS layered products listed in Table 3–1. Layered products not listed may be installed normally. Please refer to the *System Software Order Table* (SPD 28.98.xx) for the latest versions of these layered products and the processor configurations supported by each.

1. Log in to the SYSTEM account.
2. Set SYS$UPDATE as the default directory, using the following command:

   ```
   $ SET DEFAULT SYS$UPDATE
   ```

3. Using a text editor, edit the file LIBRARY.TLR and remove the following line:

   ```
   [SYSEXE]SYS.STB
   ```

4. Exit from the text editor, thus creating a new version of the file.
5. Using the text editor, edit the file REQUIRED.TLR and add the following line:

   ```
   [SYSEXE]SYS.STB
   ```

6 Exit from the text editor, thus creating a new version of the file.

7 Invoke the Tailoring command procedure and rename the library disk file as follows:

```
$ @SYS$UPDATE:VMSTAILOR
TAILOR> DISMOUNT
TAILOR> MOUNT/WRITE DAA0:  ! where DAA0: is the library disk
TAILOR> COPY/FILE SYS.STB  ! copies SYS.STB to the system disk [SYSEXE]
                           ! using the default source and target locations
TAILOR> EXIT
$ SET DEFAULT DAA0:[SYS0.SYSEXE]  ! go to the library disk in [SYSEXE]
$ RENAME SYS.STB LIBSYS.STB       ! rename the library disk version;
                                  ! the file remains unchanged on the
                                  ! system disk.
```

8 Reset the default directory to SYS$UPDATE:

```
$ SET DEFAULT SYS$UPDATE
```

9 Install the layered product using VMSINSTAL.

This operation ensures that products requiring the system global symbol table at link time during installation will find the file in the REQUIRED file group. All files that are not members of the REQUIRED file group are tailored to the library disk by VMSINSTAL during installation. The system disk is restored to its original configuration upon completion of the VMSINSTAL command procedure.

Step 7 above ensures that VMSINSTAL does not find SYS.STB on the library disk and prevents its subsequent forced removal from the system disk to save space, which would cause the installation to fail. Renaming the file on the library disk allows you to maintain a backup copy.

Any "File not found" messages that occur during installation of a software layered product may be corrected by repeating the previously listed steps to move the file to the system disk.

Please see Section 3 of the *VAX/VMS System Manager's Reference Manual* for additional information.

Table 3–1 Optional Software Products Requiring the Editing Operation for Installation on a Tailored System

ALL–IN–1
DRB32 VMS DRIVER AND UTILITIES
DRX11–C VMS DRIVER
IEX–VMS–DRIVER
MUX200/VAX
VAX COMMON DATA DICTIONARY
VAX DR11–C DRIVER
VAX DRE11–C DEVICE DRIVER
VAX DRIVER FOR 11C03
VAX NTR
VAX PRODUCER
VAX PRODUCER INTERPRETER
VAX SPM
VAX–11 TSU05 DEVICE DRIVER
VS11–VAX DRIVER
VS5XX DMA DRIVER

3.2.4 Permanent MONITOR Server Processes

Creating permanent MONITOR server processes on each member node in a cluster at bootstrap time can significantly reduce server startup time.

To create such a process, add the following lines to the appropriate startup command files. Ensure that you allot a page file quota of at least 10000 pages.

```
$ DEFINE /SYSTEM /EXECUTIVE_MODE VPM$SERVER_LIVE TRUE
$ RUN /DETACH /PAGE_FILE=10000 SYS$SYSTEM:VPM.EXE
```

You can also issue these commands interactively at any time. In this case, however, you require the following privileges: ALTPRI, NETMBX, PSWAPM, SYSNAM, SYSPRV, and TMPMBX.

3.2.5 Documentation Correction: X.25 Packet Level (Class 7) Events

In Section A.4.6 of the *VAX/VMS Network Control Program Reference Manual*, "X.25 Packet Level Events", the events numbered 7.3 through 7.14 should in fact be numbered 7.0 through 7.11, respectively.

3.2.6 Documentation Correction: Setting Up Queues for Spooled Line Printers

In Section 9.7.12 of the *VAX/VMS System Manager's Reference Manual*, "Guidelines for Setting Up Queues for Spooled Line Printers", Figures 9–6, 9–7, and 9–8 contain commands that do not conform to the procedure described in the preceding text.

The preferred method for setting up queues for spooled line printers is described in the text and not in the figures. This inconsistency will be eliminated in the next revision of the manual.

3.2.7 Documentation Correction: Creating a Command Procedure to Boot Standalone BACKUP from an Alternate System Root

Please replace the text and table found in list item 5 on page 4–41 of the Version 4.4 *Guide to VAX/VMS Software Installation* with the following text:

> On the line that begins with either the command DEPOSIT R5 or D/G 5, change the left-most digit of the number following this command to an E. For example, on a VAX-11/782, change the line that says DEPOSIT R5 4000nnnn, where the n's represent hexadecimal digits, to DEPOSIT R5 E000nnnn. On a VAX 8200, change the line that says D/G 5 0nnnnnnn to D/G 5 Ennnnnnn, where the n's represent hexadecimal digits. A different procedure, which is described in the *VAX 8800/8700/8550/8500 Console User's Guide*, is used for the VAX 8800/8700/8550/8500 processors.

3.2.8 Installing Optional Software Products: Use VMSINSTAL Instead of VMSUPDATE

The VMSUPDATE command procedure, described in Appendix C of the Version 4.4 *Guide to VAX/VMS Software Installation*, should *not* be used to install optional software products.

Instead, use the VMSINSTAL command procedure, described in Chapter 5 of the *Guide to VAX/VMS Software Installation*.

VMSUPDATE is not supported on tailored systems, cluster configurations that share a common system disk, or the VAX 8600, VAX 8650, VAX 8800, VAX 8700, VAX 8550, and VAX 8500 processors.

3.2.9 Restriction on Dual-Ported Non-DSA Disks in a VAXcluster

The following note appeared in the *VAX/VMS Release Notes, Version 4.2* and was inadvertently omitted from subsequent volumes of the release notes.

Do not use SYSGEN to AUTOCONFIGURE or CONFIGURE a dual-ported, non-DSA disk which is already available on the system via an MSCP server. Establishing a local connection to the disk when a remote path is already known will create two uncoordinated paths to the same disk. Use of these two paths will potentially corrupt files and data on any volume mounted on the drive.

In a VAXcluster, dual-ported non-DSA disks (MASSBUS or UNIBUS) may be connected between two nodes of the cluster. These disks may also be made available to the rest of the cluster using the MSCP server on either or both of the hosts to which a disk is connected.

During a normal bootstrap operation, the local path to the disk is discovered before the MSCP server path from the other host is found. If the documented restrictions regarding device naming conventions, allocation class, and the /DUAL_PORT characteristic have been observed, then the disk class driver correctly interprets the two paths as belonging to the same physical disk drive.

If the local path to the disk is not found during the bootstrap, then the MSCP server path from the other host will be the only available access to the drive. The local path will not be found during a boot if any of the following conditions exist:

1. The port select switch for the drive is not enabled for this host.
2. The disk, cable, or adapter hardware for the local path is broken.
3. There is sufficient activity on the other port to "mask" the existence of the port.
4. The system is booted in such a way that the SYSGEN AUTOCONFIGURE ALL command in the SYS$SYSTEM:STARTUP.COM procedure was not executed.

Use of the disk is still possible through the MSCP server path.

Once the configuration of the disk has reached this state, it is important *not* to add the local path back into the system I/O database. Since there is no automatic method for this to occur in VAX/VMS, the only possible way that this could occur would be to use the SYSGEN utility to AUTOCONFIGURE or CONFIGURE the device. SYSGEN is currently not able to detect the presence of the disk's MSCP path, and will incorrectly build a second set of data structures to describe it. Subsequent events could lead to incompatible and uncoordinated file operations which might corrupt the volume.

In order to recover the local path to the disk, it is necessary to reboot the system connected to that local path.

Note that if the disk is *not* dual-ported or is *never* MSCP served on the remote host, this restriction does not apply.

3.2.10 Diskette Devices and the MSCP Server

The Mass Storage Control Protocol (MSCP) does not allow all the functions associated with a diskette device. Therefore, the MSCP Server (which is based upon MSCP) will not allow diskette devices such as the RX01 and RX02 to be served. This note supplements information contained in the *VAX/VMS DCL Dictionary* and *VAX/VMS System Manager's Reference Manual.*

3.2.11 DMB32 Layered Product Software Required for DMB32 Communications Controller

VAX 8300/8200 and VAX 8800/8700/8550/8500 systems that include the DMB32 communications controller must install the DMB32 layered product in order to use the controller's synchronous port.

The VAX/VMS product kit does not contain the DMB32 software.

3.2.12 Protection of Security Auditing Information

Because all of the security auditing information is contained within the operator log file (SYS$MANAGER:OPERATOR.LOG), it is possible to lose auditing information if the disk on which this file resides becomes full.

While a well managed system will normally have adequate storage capacity, unexpected circumstances may cause excessive consumption of disk space. Should all free blocks on the disk be exhausted, a situation may arise where audit data could be lost. The National Computer Security Center (NCSC) has requested, as part of the evaluation of the VAX/VMS operating system, that a warning be issued whenever this condition occurs.

The NCSC requirement is that a message be issued prior to any audit data being lost and in sufficient time to allow the corrective action to be taken before all free blocks are exhausted.

To honor this requirement, DIGITAL is supplying the following procedure for users who wish to operate VMS as a Class C2 evaluated system. This procedure samples the available free blocks at a specified interval. The default sampling interval is every ten minutes. If the free space on the disk is less than a specified threshold, a warning messages will be issued to all terminals which have been enabled as operator terminals via a REQUEST command. The default threshold is 1% of the maximum available blocks.

This command procedure, while fully functional, is provided as a guideline to be tailored to your specific requirements.

```
$ !             SYS$MANAGER:AUDIT_GUARD.COM
$ !
$ !     Procedure to protect the audit trail when the system disk is
$ !     approaching capacity.
$
$ ! User adjustable parameters.  If no parameters are specified on the
$ ! command line, supply the default values.
$
$ IF P1.EQS."" THEN $ P1 = "00:10"
$ IF P2.EQS."" THEN $ P2 = 1
$
$ INTERVAL = P1     ! Sample remaining disk space at 10-minute intervals
$ THRESHOLD = P2    ! Report shortage when 1% of disk blocks are free
$
$ ! Determine the parameters for the device on which the operator log
$ ! file is located.
$
$ SET PROCESS/PRIVILEGE=OPER
$ LOG_FILE = F$SEARCH("SYS$MANAGER:OPERATOR.LOG")    ! For search lists
$ IF LOG_FILE.EQS."" THEN $ GOTO NO_LOG_FILE
$
$ AUD_DEV = F$PARSE (LOG_FILE,,,"DEVICE","NO_CONCEAL")
$ MAX_BLOCKS = F$GETDVI (AUD_DEV,"MAXBLOCK")
$ FREE_BLOCK_LIMIT = (MAX_BLOCKS * THRESHOLD)/100
$
$ ! Sit in a loop, checking the amount of available free space.
$
$ C2_LOOP:
$ REMAINING = F$GETDVI(AUD_DEV,"FREEBLOCKS")
$ IF (REMAINING .GT. FREE_BLOCK_LIMIT) THEN $ GOTO PAUSE
$
$ ! If the amount of free space drops below the selected threshold, report
$ ! the condition.
$
$ REQUEST  "ONLY ''REMAINING' BLOCKS AVAILABLE ON AUDIT TRAIL DISK!"
```

```
$ REQUEST  "PLEASE TAKE CORRECTIVE ACTION!"
$
$ ! Wait before checking the free space.
$
$ PAUSE:
$ WAIT 'INTERVAL'       ! Wait the interval before looking again.
$ GOTO C2_LOOP          ! Time to look again
$
$ ! If there is no log file, report it, and then exit cleanly.
$
$ NO_LOG_FILE:
$ REQUEST  "NO AUDITING INFORMATION KEPT DUE TO MISSING OPERATOR LOG FILE"
$
$ EXIT
```

This command procedure may be edited to ensure the operation is appropriate for the specific environment. The messages may be redirected to a specific device (for example, OPA0:) by use of the REPLY command or to a specific operator function by using the /TO= switch with the REQUEST command. You may also want to alter the sampling rate and threshold used by this procedure. There are two parameters for doing this:

INTERVAL	Delta time used to control the sampling rate for checking the remaining disk space and issuing the warning message.
THRESHOLD	Value representing a percentage of free blocks remaining on the disk relative to the total available blocks. Warning messages are generated if the percentage of free blocks remaining on the disk falls below this value.

If you thus modify and run the command procedure, messages will be output at the sampling rate specified by the *INTERVAL* parameter until action has been taken to increase the number of available free blocks above the *THRESHOLD* value.

Once started, this procedure will continue to execute until it is either stopped by a privileged user or the system is rebooted.

To ensure that this procedure is executed every time the system reboots, it will be necessary to add the following command to your site-specific startup command file (SYS$MANAGER:SYSTARTUP.COM):

```
$ SUBMIT SYS$MANAGER:AUDIT_GUARD /NOLOG
```

3.2.13 Documentation Correction: DTE States, Substates, and State Transitions

The following two tables correct information presented in Tables NCP–6 and NCP–7 (pages NCP–178 and NCP–179, respectively) in the *VAX/VMS Network Control Program Reference Manual*. Change bars indicate new or corrected information.

Table 3–2 (NCP–6) DTE States and Substates

State	Substate	Meaning
OFF	RUNNING	X.25 level 2 and level 3 software is operational but the DTE is not available for use.
	SYNCHRONIZING	X.25 level 2 software is operational but level 3 software is not. The DTE is not available for use. Incoming calls are cleared.
	UNSYNCHRONIZED	X.25 levels 2 and 3 are not operational and the DTE is not available for use.
ON	RUNNING	The DTE is available for normal use.
	SYNCHRONIZING	X.25 level 2 software is operational, level 3 software is starting up and the DTE will soon be available for use.
	UNSYNCHRONIZED	X.25 level 2 software is starting up and the DTE will soon be available for use.
SHUT	RUNNING	X.25 levels 2 and 3 are operational but the DTE is not to be used for any new activity; that is, all existing virtual circuits will be allowed to complete their operations.
	SYNCHRONIZING	X.25 level 2 software is operational and level 3 software is starting up. When the DTE is available for use, all existing virtual circuits will be allowed to complete their operations. Incoming calls are cleared.
	UNSYNCHRONIZED	X.25 level 2 software is starting up. When the DTE is available for use, all existing virtual circuits will be allowed to complete their operations.

Table 3–3 (NCP–7) DTE State Transitions

Old State	New State	Cause of Change
OFF-RUNNING	ON-RUNNING	Operator command: SET MODULE X25-PROTOCOL DTE STATE ON
	OFF-SYNCHRONIZING	X.25 level 3 software is resynchronizing.
	OFF-UNSYNCHRONIZED	X.25 level 2 software is resynchronizing.
OFF-UNSYNCHRONIZED	ON-UNSYNCHRONIZED	Operator command: SET MODULE X25-PROTOCOL DTE STATE ON
	OFF-SYNCHRONIZING	X.25 level 2 startup has completed.

Table 3–3 (Cont.) (NCP–7) DTE State Transitions

Old State	New State	Cause of Change
OFF-SYNCHRONIZING	ON-SYNCHRONIZING	Operator command: SET MODULE X25-PROTOCOL DTE STATE ON
	OFF-RUNNING	X.25 level 3 startup has completed.
	OFF-UNSYNCHRONIZED	X.25 level 2 software is resynchronizing.
ON-RUNNING	OFF-SYNCHRONIZING	Operator command: SET MODULE X25-PROTOCOL DTE STATE OFF
	SHUT-RUNNING	Operator command: SET MODULE X25-PROTOCOL DTE STATE SHUT
	ON-SYNCHRONIZING	X.25 level 3 software is resynchronizing.
	ON-UNSYNCHRONIZED	X.25 level 2 software is resynchronizing.
ON-UNSYNCHRONIZED	OFF-UNSYNCHRONIZED	Operator command: SET MODULE X25-PROTOCOL DTE STATE OFF
	SHUT-UNSYNCHRONIZED	Operator command: SET MODULE X25-PROTOCOL DTE STATE OFF
	ON-SYNCHRONIZING	X.25 level 2 startup has completed.
ON-SYNCHRONIZING	OFF-SYNCHRONIZING	Operator command: SET MODULE X25-PROTOCOL DTE STATE OFF
	SHUT-SYNCHRONIZING	Operator command: SET MODULE X25-PROTOCOL DTE STATE SHUT
	ON-RUNNING	X.25 level 3 startup has completed.
	ON-UNSYNCHRONIZED	X.25 level 2 software is resynchronizing.
SHUT-RUNNING	OFF-RUNNING	Operator command: SET MODULE X25-PROTOCOL DTE STATE OFF

Table 3–7 (Cont.) DTE State Transitions

Old State	New State	Cause of Change
	ON-RUNNING	Operator command: SET MODULE X25-PROTOCOL DTE STATE ON
	SHUT-SYNCHRONIZING	X.25 level 3 software is resynchronizing.
	SHUT-UNSYNCHRONIZED	X.25 level 2 software is resynchronizing.
SHUT-UNSYNCHRONIZED	OFF-UNSYNCHRONIZED	Operator command: SET MODULE X25-PROTOCOL DTE STATE OFF
	ON-UNSYNCHRONIZED	Operator command: SET MODULE X25-PROTOCOL DTE STATE ON
	SHUT-SYNCHRONIZING	X.25 level 2 startup has completed.
SHUT-SYNCHRONIZING	OFF-SYNCHRONIZING	Operator command: SET MODULE X25-PROTOCOL DTE STATE OFF
	ON-SYNCHRONIZING	Operator command: SET MODULE X25-PROTOCOL DTE STATE ON
	SHUT-RUNNING	X.25 level 3 startup has completed.
	SHUT-UNSYNCHRONIZED	X.25 level 2 software is resynchronizing.

3.3 Application Programmer Information

This section describes problems resolved in VAX/VMS Version 4.5, lists known restrictions, and contains other information about the release of interest to the application programmer.

3.3.1 Correction to $GETLKI System Service

The $GETLKI system service does not report user buffer overflow as documented for item codes LKI$_LOCKS, LKI$_BLOCKEDBY, and LKI$_BLOCKING in the *VAX/VMS System Services Reference Manual*. When used with an item descriptor specifying any of these item codes, $GETLKI will *not* set bit 31 of the **return length address** in the item descriptor when the user-supplied buffer is too small to hold the requested data.

Before Version 4.5, the LKI$_LOCKS item code, when used on a locally mastered lock, caused the entire **return length address** field to become invalid. Version 4.5 partially corrects this behavior, so that LKI$_LOCKS works as documented, *except* that bit 31 is not set as in the case described above. The result is that all of the three $GETLKI item codes that return lengthy information (LKI$_LOCKS, LKI$_BLOCKEDBY, and LKI$_BLOCKING) behave in the same (incorrect) fashion.

To work around the problem that bit 31 of the **returned length address** will never indicate when the user-supplied buffer is too small, DIGITAL suggests the following rule:

> The user buffer size *may* be inadequate if the sum of the low word of the returned length and the high word of the returned length is greater than the size of the user buffer supplied.

DIGITAL expects to correct this remaining problem in a future release.

3.3.2 VAXBI Port Communications Controller: Error Reported While Booting Standalone BACKUP

When booting Version 4.5 standalone BACKUP, a system using the VAXBI port communications controller option will display the following warning message:

```
%SYSGEN-W-NOSUCHFIL, file not found PBDRIVER.EXE
```

You can ignore this message.

3.3.3 SS$_NOENTRY Error Reported in XABPRO Block for ACL-Protected Files

In Versions 4.4 and 4.5 of VAX/VMS, files that have access control lists (ACLs) associated with them will get a status return of SS$_NOENTRY in the XAB$L_ACLSTS field of the XABPRO block if this XABPRO block was associated with the file when it was opened.

There are two suggested workarounds:

1. Ignore the status and check the size of the access control list (XAB$W_ACLLEN) returned by RMS to ensure that the user buffer size was large enough to contain the ACL.
2. Perform a $DISPLAY operation after opening the file. This will correctly return the status in XAB$L_ACLSTS.

This problem will be corrected in a future release of the operating system.

3.3.4 Ethernet/802 Drivers: Promiscuous Mode Change Planned

Software that currently utilizes the promiscuous mode feature of the Ethernet/802 drivers may need to be modified to run properly on future releases of the operating system. (See Section 6 of the *VAX/VMS I/O User's Reference Manual: Part I* for a description of these drivers.) Since the Ethernet/802 drivers now allow a wide variety of packets to be transmitted and received, some restrictions will be placed on channels that turn on the promiscuous mode (NMA$C_PCLI_PRM) parameter. When this parameter is turned on, the following rules will apply:

- Both Ethernet and IEEE 802 formatted packets will be received. The P5 buffer, if specified, must be at least 16 bytes long.
- Only one type of packet may be transmitted: either Ethernet or IEEE 802. The value of the NMA$C_PCLI_FMT parameter will be used to determine which format will be used for transmissions.
- The NMA$C_PCLI_PAD parameter will be ignored during READ operations on channels that are running in promiscuous mode.
- The promiscuous mode channel may not be put into SHARED access mode. Attempts to put the promiscuous mode channel into shared mode will result in an SS$_BADPARAM error.

Applications using the promiscuous mode feature should note these planned restrictions. Those applications should be modified to run within these restrictions before the restrictions are applied and shipped in the VMS Ethernet/802 drivers.

3.3.5 Ethernet Controller: List of Expected Errors

Certain Ethernet controllers support features that allow them to communicate with the hardware outside the VAX system to detect hardware failures. If the hardware connected to the Ethernet controller does not support these "hardware failure detection" features, then the controller and driver will report errors which are not true errors.

To facilitate the detection of true errors, use the following list of "expected" errors to eliminate those errors which are caused by the lack of hardware failure detection support.

- When using the DEQNA with the DECOM transceiver, a "Send failure" with reason code "Short circuit" will be reported for each packet transmitted.
- When using the DEUNA or DELUA with broadband, a "Collision detect check failure" will be reported for each packet transmitted.

See Section 6 of the *VAX/VMS I/O User's Reference Manual: Part II* for a discussion of these controllers.

3.3.6 Documentation Correction: Screen Management Routines Using AST Routines

In the *VAX/VMS Run-Time Library Routines Reference Manual*, the descriptions of the screen management routines SMG$ENABLE_UNSOLICITED_INPUT, SMG$SET_BROADCAST_TRAPPING, and SMG$SET_OUT_OF_BAND_ASTS should describe the parameter passing mechanism for *AST-routine* as "address of entry mask by value" and not as "entry mask by reference."

3.3.7 SCNRTL Problems Corrected

Version 4.5 corrects the following VAX SCAN Run-Time Library (SCNRTL) problems:

- A PRUNE statement causing an access violation, depending on the order in which nodes have been added to the tree that is being pruned.
- A call to SCN$GET_TOKEN_NAME resulting in an access violation during virtual memory cleanup.
- Specification of VARYING STRING as the input stream resulting in data corruption, because of the length being improperly determined from the descriptor.

3.3.8 TU78 Tape Driver (TFDRIVER) Error Handling

Version 4.5 corrects a problem that occurred under certain conditions when I/O requests were issued to a TU78 tape drive.

The problem occurred when there was an attempt to read a record that could not be read in the forward direction. The tape would apparently loop forever, despite the TU78 driver (TFDRIVER.EXE) returning a successful status (SS$_NORMAL).

When there is an attempt to read such a record, the TM78 tape controller repositions the tape to the beginning of the record and returns a status to the driver indicating that it retry the READ operation in the forward direction. After several retries and repositions, the TM78 controller does not reposition the tape (leaving it at the end of the problem record). At this time, the controller returns a status to the driver indicating that it should attempt READ REVERSE. This happens several times, with the TM78 repositioning the tape to the end of the record after each failure to read in the reverse direction. If any of the READ REVERSE operations is successful in reading the record, then the tape is left positioned at the beginning of the record. Starting at the beginning of the record, the next READ command once again attempts to read this same record.

3.3.9 TE16 and TU77 Tape Driver (TMDRIVER) Corrections

Prior to Version 4.5, two problems were evident in TMDRIVER:

- The MT$M_EOT bit was not always set in UCB$L_DEVDEPEND when the drive detected the EOT mark.
- The MT$M_EOF bit did not always appear to be set in UCB$L_DEVDEPEND when the tape was positioned at a tape mark. (This problem existed only for 1600-bpi tapes.)

VAX/VMS Version 4.5 corrects both of these conditions.

3.3.10 Caution on Use of NOP Instruction as a Delay Mechanism

DIGITAL recommends that you do *not* use the VAX MACRO NOP (No Operation) instruction as a means of delaying program execution.

The delay time caused by the NOP instruction is dependent on processor type. For instance, the VAX 8600, VAX 8650, VAX 8800, VAX 8700, VAX 8550, or VAX 8500 processors, execute the NOP instruction more quickly than other VAX processors.

Whenever you must have a program wait for a specified time period, you should use a macro or code sequence that is not dependent on the processor's internal speed. For example, you can use the TIMEWAIT macro, which is documented in the *Writing a Device Driver for VAX/VMS* volume. You can also use the Set Timer ($SETIMR) and Wait for Single Event Flag ($WAITFR) system services, as described in the *VAX/VMS System Services Reference Manual*, to force such delays.

3.3.11 Debugging Shareable Images—Change in Behavior from Pre-Version 4.4 Releases

If, prior to Version 4.4., you linked your shareable images using the LINK/SHARE command, you should be aware that linking shareable images in this manner will now result in traceback information being passed to the shareable image.

When you debug your program, and execution is suspended within that shareable image, the debugger will set the image automatically. This is called dynamic image setting.

This will result in different symbolic information being made available. For example, the display for SHOW CALLS will look different. In contrast to module setting, the symbol information for only the currently set image is available at any one time. (See the description of the SET IMAGE, SET MODULE, and SET MODE [NO]DYNAMIC commands in the *VAX/VMS Debugger Reference Manual* for more information).

If you prefer the old behavior, you can link your shareable image with the command LINK/SHARE/NOTRACE. Traceback information will not be present in the image and DEBUG will not set the image.

To take full advantage of the new shareable image support, you should link your shareable image with the command LINK/SHARE/DEBUG. Then full symbol table information will be available, the debugger will set the image, and you can perform symbolic debugging of that shareable image.

3.3.12 Failure of VAX BASIC SET INITIAL CHOICE Statement

In VAX BASIC Version 3.0, the SET INITIAL CHOICE statement will fail when it is used with a choice array with no count specified. To avoid this problem, specify a count clause when you use this statement.

This problem will be corrected in a future software release.

3.3.13 SYS$CREMBX and Process-Private Logical Names

Version 4.4 introduced a change to the behavior of logical names that are automatically created as a part of mailbox creation or when a volume is mounted. Specifically, if a logical name table has been redirected to point to a process-private name table, the logical name is no longer deleted when the mailbox disappears or the volume is dismounted. (See page 2–21 of the *VAX/VMS Release Notes, Version 4.4* for details of this change.)

Version 4.5 completes the decoupling of associated logical names from their creator when the names are placed into a process-private table. This change may affect a small number of programs that use the Create Mailbox system service (SYS$CREMBX).

Programs will only be affected under the following set of circumstances.

- The associated logical name table (LNM$TEMPORARY_MAILBOX or LNM$PERMANENT_MAILBOX) has been redefined to point to a process-private table.
- The program issues several $CREMBX calls from several threads of execution, using the same logical name.

The new behavior is that each call to $CREMBX under these circumstances will cause a new mailbox unit to be created and assigned to a new channel.

Note that applications that place mailbox names into shared name tables are unaffected by this change. That is, the second and succeeding calls to $CREMBX will assign new channels to the existing mailbox unit.

Note further that applications using several cooperating processes are also unaffected, even if the names are placed into a process-private table. (If the names existed in a process-private table, they were invisible to other processes even before this change was made.)

3.3.14 DECnet–VAX File Operations with ULTRIX-32

VAX/VMS support for file operations to ULTRIX systems supports only sequential stream-lf files. Variable-length record format files are converted to stream-lf as they are transmitted. However, stream and stream-cr files are not sent, and return a "network operation not supported" error.

Unfortunately, files copied from an ULTRIX system are created in stream format, and cannot be copied back. A workaround for this problem is to use the CONVERT utility to convert stream files to stream-lf format. This may be done on the VAX/VMS system, or in copying the file. For example:

```
$ CONVERT/FDL=STREAM_LF STREAM_FILE ULTRIX"account password"::"stream_lf_file"
```

where STREAM_LF.FDL contains:

```
TITLE   "stream_lf"
IDENT   "20-AUG-1986 11:46:53   VAX-11 FDL Editor"
SYSTEM
        SOURCE                          VAX/VMS
        FILE
                ORGANIZATION            sequential
        RECORD
                BLOCK_SPAN              yes
                CARRIAGE_CONTROL        carriage_return
                FORMAT                  stream_LF
                SIZE                    0
```

3.4 System Programmer Information

This section describes problems resolved in VAX/VMS Version 4.5, lists known restrictions, and contains other information about the release of interest to the system programmer.

3.4.1 NETACP Verification of MOP Messages

Version 4.5 of the network ancillary control program (NETACP) performs some verification before it starts up a maintenance operation module (MOM) process to service an incoming maintenance operation protocol (MOP) request. (The *VAX/VMS Networking Manual* discusses these topics.)

NETACP will only start up a MOM process under the following conditions:

- The request is not directed at a multicast address.
- The source node specified in the MOP request is defined in NETACP's node database.
- The MOP message requests an operating system and contains the software identification of the file to be loaded.
- The MOP request is not for a load or dump.

If NETACP does not start up a MOM process, it will generate an event message of type 0.7 (aborted service request, Line open error). The Ethernet address of the source node will also be displayed with the message.

3.4.2 Documentation Correction: IFNORD, IFNOWRT, IFRD, and IFWRT Macros

The descriptions of the IFNORD, IFNOWRT, IFRD, and IFWRT macros in pages B–16 through B–19 of the *Writing a Device Driver for VAX/VMS* volume erroneously define the **dest** argument in each case.

The published information is incorrect in both the text and parameter definition list. The following table supplies the correct definitions:

Macro	Definition of "dest"
IFNORD	Address to which IFNORD passes control if either of the specified bytes cannot be read in the specified access mode
IFNOWRT	Address to which IFNOWRT passes control if either of the specified bytes cannot be written in the specified access mode
IFRD	Address to which IFRD passes control if both bytes can be read in the specified access mode
IFWRT	Address to which IFWRT passes control if both bytes can be written in the specified access mode

3.4.3 Behavior of Zero-Length and Negative Byte Counts Submitted in $QIO Requests

Version 4.4 introduced a change into function-decision table (FDT) routines that prevented negative byte counts from being passed to the Queue-I/O Request system service (SYS$QIO) and its support subroutines. This check also disallowed byte counts of zero. While this change did not affect any drivers that are part of the VMS kit, it may have caused problems for user-written drivers.

Version 4.5 relaxes the restriction. While negative byte counts still cause an error return of SS$_BADPARAM, zero-length byte count transfers are again allowed.

3.4.4 Documentation Correction: Bootstrapping with XDELTA on VAX 8700, VAX 8550, VAX 8500, and VAX 8300 Systems

In Section 15.1 of the *Writing a Device Driver for VAX/VMS* manual, the instructions for bootstrapping with XDELTA on the VAX 8200 and VAX 8800 systems apply as well to the VAX 8300, VAX 8500, VAX 8550, and VAX 8700 systems.

3.4.5 Documentation Correction: EXE$QIODRVPKT

The executive routine EXE$QIODRVPKT was inadvertently omitted from Appendix C of the Version 4.4 *Writing a Device Driver for VAX/VMS* manual. (It is, however, described in Section 8 of that volume.)

The following is the routine description:

EXE$QIODRVPKT

Module: SYSQIOREQ

Driver FDT routines call EXE$QIODRVPKT to send an IRP to a driver start-I/O routine. This routine calls EXE$INSIOQ and then transfers control to EXE$QIORETURN.

input

Registers	Contents
R3	Address of IRP for the current I/O request
R4	Address of current PCB
R5	Address of UCB
Fields	**Contents**
UCB$B_FIPL	Driver fork IPL
UCB$V_BSY (in UCB$L_STS)	Unit busy flag
UCB$L_IOQFL	Address of unit I/O queue listhead

3.4.6 Documentation Correction: $DEF Macro

Page B–5 of the *Writing a Device Driver for VAX/VMS* volume incorrectly describes the method for defining a second symbolic name for a single field. The following text should appear as the second paragraph in the description of the **alloc** argument:

> You can define a second symbolic name for a single field, using the $DEF macro a second time immediately following the first definition, leaving the **alloc** argument blank in the first definition. The following example does this, equating SYNONYM2 with LABEL2:

```
$DEFINI JLB                     ; Start structure definition
$DEF    LABEL1  .BLKL  1        ; First JLB field
$DEF    SYNONYM2                ; Synonym for LABEL2 field
$DEF    LABEL2  .BLKL  1        ; Second JLB field
$DEF    LABEL3  .BLKL  1        ; Third JLB field
$DEFEND JLB                     ; End of JLB structure
```

A VAX/VMS Version 4.5 Update Description

This appendix contains a listing of the patches, new images, and miscellaneous fixes contained in Version 4.5 update kit. This listing is obtained from the text file SYS$UPDATE:VMS045.TXT that is produced by the installation procedure if option 2 or 3 is selected, as described in step 4 of Section 1.4.

```
1) ADARTL (patch image)

  ! ADARTL.EXE
  !
  !   ECO01   SBL             23-Apr-1986
  !           MODULE: ADA$END_OF_FILE (X-3)
  !           SPR: 11-87184
  !           An uninitialized variable prevented END_OF_FILE from
  !           returning a consistent value when used in packages
  !           DIRECT_IO or DIRECT_MIXED_IO.

2) AGEN (miscellaneous fix)

  ! AUTOGEN.VUG
  !
  !   ECO01   GHC0002         23-May-1986
  !           MODULE: AUTOGEN
  !           Set up to use the appropriate AUTOGEN update file based on
  !           which version of AUTOGEN currently exists on the system.

3) AUTOGEN (edit text file)

  ! AUTOGEN.COM
  !
  !   ECO01   GHC0005         12-May-1986
  !           MODULE: AUTOGEN
  !           Allow ADD_, MIN_, MAX_ symbols in MODPARAMS.DAT for any
  !           numeric SYSGEN parameter. Start the MSCP server early in
  !           the boot cycle.

4) BACKUP (patch image)

  ! BACKUP.EXE
  !
  !   ECO006  KGW00021        30-May-1986
  !           MODULE: TAPEUTIL
  !           Patch MUST be applied to VMS V4.4 systems ONLY.
  !
  !           Excessive parity errors on a TA78 would result in
  !           access to the tape being lost.
  !           Repair this by unloading the tape if the RESTART
  !           option is taken from the error handler.

5) BASRTL (miscellaneous fix)

  !  BASRTL.EXE
```

6) BASRTL (patch image)

```
! BASRTL.EXE
!
!   EC001   JCW1003         04-Mar-1986
!           MODULE: BAS$POWHH
!           Fix SPR 11-84761.   Routine sometimes incorrectly
!           returned a negative result.
!
!           KC2009          07-Apr-1986
!           MODULE: BAS$CTRLC
!           Clear ASTs only if the user is trapping CTRL/Cs.
!
!           KC2011          18-Apr-1986
!           MODULE: BAS$ERROR
!           Change OPTION HANDLE semantics.
!
!           KC1119          20-Apr-1986
!           MODULE: BAS$OPEN
!           OPEN_HANDLER should free the wildcard context that
!           gets allocated.
!
!   EC002   KC1080          15-May-1986
!           MODULE: BAS$$UDF_WL
!           Fix SPR 11-87232.  PRINT 1,2,3,4,5,6,7,8,9,10 correctly
!           does not repeat the ninth element.
!
!           KC1019          15-May-1986
!           MODULE: BAS$$EXIT_HANDL
!           PUTMSG and UNWIND instead of resignaling conditions.
!
!           KC1012          15-May-1986
!           MODULE: BAS$$PUR_IO_BUF
!           If the RAB ISI is zero, do not try to purge the
!           dirty buffer.
!
!           KC1119          16-May-1986
!           MODULE: BAS$OPEN
!           Fix quad-key test.
!
!   EC003   KC1063          19-May-1986
!           MODULE: BAS$CVT_OUT
!           Fix PRINT USING and FORMAT$ packed decimal bug.
!
!   EC004   KC2012          16-Jun-1986
!           MODULE: BAS$ERROR
!           Fix an ON ERROR GO BACK bug.  Fix handling of
!           non-BASIC errors.
!
!           KC1120          24-Jun-1986
!           MODULE: BAS$OPEN
!           Fix OPEN_HANDLER so it does not destroy the
!           expanded string name.
```

```
7) BASRTL2 (patch image)

  ! BASRTL2.EXE
  !
  !   EC0001  Bundled fixes for V4.5.
  !
  !           KC1010          25-Feb-1986
  !           MODULE: BAS$GR_OUTPUT_MISC
  !           Complain if the user specified one point, did not have a semicolon,
  !           and the beam was not previously on.
  !
  !           KC1005          25-Feb-1986
  !           MODULE: BAS$GR_INIT_INP
  !           For choice, verify that the count specified is within the size
  !           of the array.
  !
  !           KC1006          25-Feb-1986
  !           MODULE: BAS$GR_SET_ECHO_AREA
  !           For choice, call the routine that handles INQ_CHOICE_STATE.
  !
  !           KC1008          27-Feb-1986
  !           MODULE: BAS$GR_ASK_ECHO_AREA
  !           For choice, call the routine that handles INQ_CHOICE_STATE.
  !
  !           KC1015          02-Mar-1986
  !           MODULE: BAS$GR_ASK_MISC
  !           Change ASK TEXT ANGLE formula.
  !
  !           KC1010          02-Mar-1986
  !           MODULE: BAS$GR_ASK_CAP
  !           ASK MAX COLOR should return NPREIDX - 1.
  !
  !           KC1019          02-Mar-1986
  !           MODULE: BAS$MAT_ASSIGN
  !           Correct conversion to double precision to handle scaling
  !           properly.
  !
  !   EC0002  More bundled fixes for V4.5.
  !
  !           KC1007          06-Mar-1986
  !           MODULE: BAS$GR_INIT_INP
  !           Fix earlier patch for edit KC1007 to reflect the source fix.
  !
  !           KC1011          06-Mar-1986
  !           MODULE: BAS$GR_ERROR
  !           In SCAN_ERROR, only scan the list if the list was specified.
  !
  !   EC0003  More bundled fixes for V4.5.
  !
  !           KC1009          15-Mar-1986
  !           MODULE: BAS$GR_ASK_ECHO_AREA
  !           For string input, make sure the string descriptor has a nonzero
  !           pointer.   A null string is not valid.
  !
  !           KC1019          15-Mar-1986
  !           MODULE: BAS$$GR_UTIL
  !           Change the exit handler to be a special routine that
  !           calls GKS$EMERGENCY_CLOSE.
  !
  !           KC1006          17-Mar-1986
  !           MODULE: BAS$GR_INIT_INP
  !           For point, initialize RET_SIZE to work around a GKS bug.
  !
```

```
!   EC0004  More V4.5 bundled fixes.
!
!           KC1005          19-Apr-1986
!           MODULE: BAS$INKEY
!           If we get a CTRL/C then signal CTRL/C.
!
!   EC0005  More V4.5 bundled fixes.
!
!           KC1015          15-May-1986
!           MODULE: BAS$GR_CTRL
!           CLEAR_WS should update the workstation.
!
!           KC1011          15-May-1986
!           MODULE: BAS$GR_OUTPUT
!           If the count specified is larger than the size of the arrays
!           then signal an error.
!
!           KC1007          15-May-1986
!           MODULE: BAS$GR_INIT_INP
!           If the count specified is larger than the size of the arrays, then
!           signal an error.   Also, DIM parameter should be read/write storage
!           not read-only.  Also, for valuator, check that the initial value
!           is within the range.
!
!           KC1013          15-May-1986
!           MODULE: BAS$GR_INPUT
!           LOCATE VALUE should check that the initial value is within
!           the range, if one is specified.
!
!           KC1012          16-May-1986
!           MODULE: BAS$GR_OUTPUT
!           Correctly handle cases where the user specified too few
!           coordinates.
!
!           KC1010          16-May-1986
!           MODULE: BAS$GR_SET_VIEWING
!           Fix SET_INP_PRIO.
!
!           KC1012          16-May-1986
!           MODULE: BAS$GR_ERROR
!           Translate some new messages.
!
!   EC0006  More V4.5 fixes.
!
!           KC1008          20-May-1986
!           MODULE: BAS$GR_ASK_WS
!           Check that the text extent arrays are at least four elements long.
!
!           KC1013          22-May-1986
!           MODULE: BAS$GR_ERROR
!           Translate some more messages.
!
!           KC1008          27-May-1986
!           MODULE: BAS$GR_INIT_INP
!           Correct KC1007.
!
!           KC1014          27-May-1986
!           MODULE: BAS$GR_INPUT
!           Correct KC1013.
!
```

```
!   EC0007  More V4.5 fixes.
!
!           KC1014          16-Jun-1986
!           MODULE: BAS$GR_ERROR
!           Fix the translation table.

8) CLUSTRLOA (new image)

! CLUSTRLOA.EXE
!
!
!   X-3     ROW0554         21-Mar-1986
!           MODULE: DSTRLOCK
!           Change handling of conversion-not-queued requests so that
!           repeat "deliver blocking AST" indications are ignored.  This
!           prevents duplicate deliveries of blocking ASTs.
!
!   X-5     DWT0262         14-Apr-1986
!           MODULE: ACKMSG.MAR,CONMAN.MAR,CNXMAN.MAR,CJFCLUSTR.MAR
!           Support notification of RMS journaling on removal of a node
!           from a cluster.  Remove module CJFCLUSTR.MAR.
!
!
!   X-3U1   DWT0270         15-May-1986
!           Set RSB$L_CSID field when a "new lock granted" or "waiting"
!           message is received.  This corrects a problem when
!           multiple new root lock requests are outstanding on a
!           single resource.

9) COBRTL (miscellaneous fix)

! COBRTL.EXE

10) COBRTL (patch image)

! COBRTL.EXE
!
!   EC001   MDL             01-Apr-1986
!           MODULE: COB$ACCEPT, routine COB$ACC_SCR
!           Zero FIRST_CHARS_READ when there is a conversion
!           error in the case of NO BLANK CONVERSION.  Otherwise
!           corrupt data can be returned on a reprompt.
!
!   EC002   MDL             14-Apr-1986
!           MODULE: COB$ACCEPT, routine COB$ACC_SCR
!           Reset ACC_SIZE in the same situation as above (EC001).
!
!   EC003   MDL             14-Apr-1986
!           MODULE: COB$ACCEPT, routine COB$$ILLEGAL_TERM
!           Subtract off FIRST_CHARS_READ when figuring remaining
!           number of characters to get after an illegal terminator
!           has been entered and the field is not full yet.
!
!   EC004   MDL             12-May-1986
!           MODULE: COB$ACCECV, routine COB$$NUMERIC_CONV
!           Fix stripping of trailing zeros after the decimal point.
!           This is a correction to edit 1-002, introduced in V4.4.
```

```
11) CONVSHR (patch image)

   ! CONVSHR.EXE
   !
   !   ECO1    JWT0238         23-Apr-1986
   !           MODULE: RECL$REC
   !           CONVERT/RECLAIM can leave behind partially
   !           reclaimed buckets under the same circumstances
   !           as the earlier problem discovered with the
   !           reclaiming of records with key compression.
   !           If any nondeleted record is encountered in
   !           the bucket, don't modify the bucket.

12) CTDRIVER (new image)

   ! CTDRIVER.EXE
   !
   !   ECO0001 DSS003          11-Mar-1986
   !           Add quota checking to prevent RWAST state. Add OOB
   !           INCLUDE checking.

13) CWDRIVER (new image)

   ! CWDRIVER.VUI
   !
   !   ECO01   CBD0027         02-Sep-1986
   !           Fix use of WFIKPCH. Use real timeouts and make interrupt
   !           service clear INT and TIM in UCB status.

14) CWDRIVER (patch image)

   ! CWDRIVER.EXE
   !
   !   ECO01   CBD0009         30-Apr-1986
   !           MODULE: CWDRIVER
   !           Fix typo in handling error packet from the console.
   !           Give error code if drive is write-locked rather than
   !           hanging the process doing the write.

15) DCL (new image)

   ! DCL.EXE
   !
   !   ECO01   HWS0002         16-May-1986
   !           Fix various DCL problems.

16) DCLTABLES (miscellaneous fix)

   ! DCLTABLES
   !
   !   ECO01   HWS0001         16-May-1986
   !           Update the command definition for SET.

17) DEBUG (patch image)

   ! DEBUG.EXE
   !
   !   ECO01   RT01               Apr-1986
   !           Part of the patch to turn DEBUG V4.4 into DEBUG V4.5. This ECO
   !           fixes two bugs in the shareable image support.
   !
```

```
!   EC002   RT02                Apr-1986
!           Part of the patch to turn DEBUG V4.4 into DEBUG V4.5. This ECO
!           fixes two more bugs: a bug with examining arrays whose
!           elements are larger than 2**16 bytes, and a bug in which
!           the debugger infinite-loops if an Ada program has a
!           BPT instruction in it which was not placed there by
!           the debugger.
!
!   EC003   RT03                May-1986
!           Correction to fix #1. Make sure we set the SEC$M_WRT bit as
!           well as the SEC$M_CRF bit.
!
!   EC004   RT04                May-1986
!           Fix a problem where DEBUG fails with UIS V3.0 on VAXstations
!           if the default VT220 window size is set to anything other than
!           24 by 80.
!
!   EC005   RT05                Jun-1986
!           Fix error in EC004.
!
!   EC006   RT06                Jul-1986
!           Change the method DEBUG uses to control the separate window
!           it creates on a VAXstation so that it uses the terminal
!           emulator OSC sequences. This avoids a potential VAXstation
!           hang or crash situation.
```

18) DSDRIVER (miscellaneous fix)

```
! DSDRIVER.MSKEXE
!
!           MAS0065         27-May-1986
!           Fix connection walking bug that leads to spurious host clears
!           of HSCs following virtual circuit failures during failover.
!           Correct other miscellaneous problems involving host and
!           controller timeouts.  Relevant module audit versions are
!           DUDRIVER (X-27) and DUTUSUBS (X-30).
!
!           ROW0561         12-May-1986
!           MODULE: DUSHADOW, DUMNTVER
!           Fix DU$START_REMSHAD to do identical copy-in-progress cleanup
!           when condition detected by either CPYSEQNUM check or controller
!           reported error.
!           Fix bad branch destination after IO$_REMSHAD failure test in
!           Step V.  Also, fix Step IV to tolerate an SS$_MEDOFL error from
!           the IO$_AVAILABLE ... function.
!
!           ROW0559         07-Apr-1986
!           MODULE: DUMNTVER
!           Change Step II substep K-4 to account for the fact that
!           MNTVER$FREE_BUFFER does not preserve R1 and R2.
```

19) DTKRTL (miscellaneous fix)

```
! DTKMSG.OBJ
!
!   1-002   TS                  11-Apr-1986
!           MODULE: DTK$MSGDEF
!           Add the error messages for the DTK$
!           facility to STARLET.OLB so users can
!           access them from their source code.
```

```
20) DTKSHR (new image)

    ! DTKSHR.EXE
    !
    !   1-003   TS              11-Apr-1986
    !           MODULE: DTK$UTIL
    !           Fix to an internal routine, DTK$$GET_STATUS
    !           for a timing problem with reading back a phone
    !           status from the DECtalk.

21) DUDRIVER (new image)

    ! DUDRIVER.EXE
    !
    !   EC003   MAS0065         27-May-1986
    !           Fix connection walking bug that leads to spurious host clears
    !           of HSCs following virtual circuit failures during failover.
    !           Correct other miscellaneous problems involving host and
    !           controller timeouts.  Relevant module audit versions are
    !           DUDRIVER (X-27) and DUTUSUBS (X-30).
    !
    !   EC002   PRD0226         16-May-1986
    !           MODULE: DUDRIVER
    !           Relink with new DUTUSUBS so image matches V4.5 stream.
    !
    !   EC001   ROW0560         26-Apr-1986
    !           MODULE: DUDRIVER
    !           Fix the SSM test after RECORD_UNIT_STATUS in IO$_PACKACK
    !           processing to use R3 (the UCB address) instead of R5 (the CDRP
    !           address).

22) ERFBRIEF (patch image)

    ! ERFBRIEF.EXE
    !
    !   EC001   SAR0496         17-Apr-1986
    !           MODULE: BRIEF_C_DISPATCHER
    !           Enable VAXBI port communications controller support.
    !
    !           MODULE: BRIEF_DEVICE_ENTRY_ROUTINES
    !           Enable VAXBI port communications controller support.

23) ERFBUS (new image)

    ! ERFBUS.EXE
    !
    !   EC001   SAR0500         16-Apr-1986
    !           MODULE: DECODE_PA_PB_DRIVER_ENTRIES
    !           Correct output of PMDATR and control store value.
    !           Update PESR MISC text as per new CI codes.

24) ERFCTLSHR (patch image)

    ! ERFCTLSHR.EXE
    !
    !   EC001   RAP0093         23-Apr-1986
    !           MODULE: RECSELECT.B32
    !           Currently when /BEFORE is selected and we are processing
    !           a record later than the date specified ERF exits prematurely.
    !           This patch allows ERF to continue processing subsequent records.
    !
    !   EC002   RAP0095         23-Apr-1986
    !           MODULE: ERFCONTRL.B32
    !           Initialize SYE$L_OPTIONS with 'S' before calling FULL_DISPATCHER
    !           Update the version number of ANALYZE/ERROR_LOG.
```

```
25) ERFLIB (new file)

   ! ERFLIB.TLB
   !
   !   EC001   SAR0505         15-Apr-1986
   !           MODULE: N/A
   !           Enable AIE-NI support.

26) ERFPROC1 (patch image)

   ! ERFPROC1.EXE
   !
   !   EC001   RAP0092         09-Apr-1986
   !           MODULE: MSCPTXT.B32
   !           Text change: RETRIES <-- RETRIES LEFT
   !           This field was incorrectly translated as the number of retries
   !           left.
   !
   !   EC002   SAR0503         29-Apr-1986
   !           MODULE: PADRIVER_LOGMESSAGE
   !           Update port text routine.
   !
   !   EC003   RAP0103         06-Jul-1986
   !           MODULE: RAXXDVDP.B32
   !           Only call DECODE_HSC_REQUESTOR if controller is an HSC.
   !
   !           MODULE: MSCPDTDSP.B32
   !           Change the known length of SDI format packet from 56 to 49.

27) ERFSHR (patch image)

   ! ERFSHR.EXE
   !
   !   EC001   SAR0502         29-Apr-1986
   !           MODULE: DECODE_PA_PB_SHARE
   !           Added AIE-NI support.
   !
   !   EC002   RAP0094         28-Apr-1986
   !           MODULE: LOGGER.B32
   !           Add 8700 and 8550 support.

28) ERFSHR2 (patch image)

   ! ERFSHR2.EXE
   !
   !   EC001   SAR0499         22-Apr-1986
   !           MODULE: DECODE_BIIC_REGISTERS
   !           Correct the AIE/AIO DTYPE values and the index
   !           calculation for the NSLAVE_RTN table.

29) ETDRIVER (new image)

   ! ETDRIVER.EXE
   !
   !   EC001   RBH0001         02-Apr-1986
   !           MODULE: ETDRIVER
   !           Add support for VAXBI port communications controller device.
   !
   !   EC002   RBH0002         16-May-1986
   !           MODULE: ETDRIVER
   !           Include support for IEEE 802 protocol and multiport
   !           synchronization.
```

```
30) F11AACP (patch image)

   ! F11AACP.EXE
   !
   !   EC001   ACG0521         20-Jun-1986
   !           MODULES: NONE
   !           Build a patch descriptor for the remaining unused space
   !           in the image file.
   !
   !   EC002   ACG0521         20-Jun-1986
   !           MODULES: DISPAT, IODONE
   !           Correct the dispatching of IO$_DSE (data security erase)
   !           functions when they are sent to the file system for a
   !           window turn.

31) F11BXQP (patch image)

   ! F11BXQP.EXE
   !
   !   EC001   ACG0519         08-May-1986
   !           MODULE: CREATE
   !           Correct a problem that could result in noncontiguous
   !           space allocation on Files-11 subset 0 volume sets.
   !
   !   EC002   LMP0331         11-Jun-1986
   !           MODULE: DEACCS
   !           Undo the change made in LMP0331 to enable the protection check
   !           on the write attributes call.
   !
   !   EC003,  ACG0521         20-Jun-1986
   !   EC004   MODULES: DISPAT, IODONE
   !           Correct the dispatching of IO$_DSE (data security erase)
   !           functions when they are sent to the file system for a
   !           window turn.
   !
   !   EC005   ACG0523         14-Jul-1986
   !           MODULE: DIRSCN
   !           Correct a problem that resulted in new versions of
   !           files created on a volume set sometimes acquiring
   !           incorrect protection and ownership attributes.
   !
   !   EC006   ACG0524         15-Jul-1986
   !           MODULE: DELFIL
   !           When a multivolume file is deleted, ensure that the
   !           file ID of the primary header is released on the right
   !           volume. This problem occasionally causes HDRNOTMAP
   !           bugchecks when operating on volume sets.

32) FORRTL (miscellaneous fix)

   !  FORRTL.EXE

33) FORRTL (patch image)

   ! FORRTL.EXE
   !
   !   EC001   KC1018          12-May-1986
   !           MODULE: FOR$INQUIRE
   !           Fix QAR171 from the V4.4 database.  INQUIRE should
   !           be sure to dispense with the wildcard context
   !           created for use by RMS.
```

34) JOBCTL (patch image)

```
! JOBCTL.EXE
!
!   EC001   JES0001         02-Apr-1986
!           MODULE: ACCOUNTNG
!           Remove call to CLEAR_ACCOUNTING_FLAGS from routine
!           CLOSE_ACCOUNTING_FILE to fix bug where a SET ACCOUNTING/NEW
!           would unintentionally turn off accounting.
!
!           Add a call to CLEAR_ACCOUNTING_FLAGS in routine
!           WRITE_ACCOUNTING_FILE to turn off accounting if there
!           is an error while trying to write to the accounting file.
!
!   EC002   JES002          02-Apr-1986
!           MODULE: SNDJBC
!           Add a call to CLEAR_ACCOUNTING_FLAGS to fix command
!           SET ACCOUNTING/DISABLE.  Previously this command would have
!           had no effect. Now accounting is turned on with all options
!           as stated in the documentation.
!
!   EC003   JES003          16-Apr-1986
!           MODULE: SNDJBC
!           Check status after call to ENQUEUE_JOB in routine ALTER_JOB.
!           If an error is returned do not rewrite the job record.
!
!   EC004   JES004          16-Apr-1986
!           MODULE: SCHEDULER
!           Check status after call to ENQUEUE_JOB in routine AFTER_NONAST.
!           If an error is returned do not rewrite the job record.
!
!   EC005   JES005          16-Apr-1986
!           MODULE: SYMBIONT
!           Check status after call to ENQUEUE_JOB in routine
!           PROCESS_SYMBIONT_MESSAGE.  If an error is returned do not
!           rewrite the job record.
!
!   EC006   JES006          12-May-1986
!           MODULE: EXECUTOR
!           Call FIND_PENDING_JOBS in RESUME_EXECUTION routine to fix
!           bug where jobs that could execute after a queue had been
!           started remained pending.
!
!   EC007   JES007          12-May-1986
!           MODULE: SCHEDULER
!           Call EXECUTOR_ACCEPTS_JOB a final time in FIND_AVAILABLE_EXECUTOR
!           to ensure that the job's form is correct.
!
!   EC008   JES008          21-Aug-1986
!           MODULE: CONTROL
!           Nullify use of VMSD2 parameters by clearing the
!           FLAGS_V_READ_VMSD2 bit.
```

```
35) LBRSHR (new image)

   ! SYS$LIBRARY:LBRSHR
   !
   !           ROP0084         02-Apr-1986
   !           Correct truncation of help text.

36) LCDRIVER (new image)

   ! LCDRIVER.EXE
   !
   !   ECO0001 RRB0001         03-Apr-1986
   !           Fix map allocation problem when /FALLBACK is set.

37) LIB (miscellaneous fix)

   ! LIB.MLB
   !
   !   ECO02   EJL004          13-May-1986
   !           MODULES: $ADPDEF, $BVPDEF
   !           Add VAXBI VAX Port definitions.
   !
   !   ECO01   EJL003          24-Apr-1986
   !           MODULE: $CIBDTDEF
   !           Correct CI BDT alignment.

38) LOGINOUT (patch image)

   ! LOGINOUT.EXE
   !
   !   ECO01   ACG0518         31-Mar-1986
   !           Fix setup of the extended process rights lists in
   !           the case where LOGIN is started with an existing
   !           extended rights list.

39) LPDRIVER (new image)

   ! LPDRIVER.EXE
   !
   !   ECO0001 RRB0001         03-Mar-1986
   !           Fix byte count quota problems when I/O fails.  Support uppercase
   !           characters for open and close brace characters.

40) MAIL (patch image)

   ! MAIL.EXE
   !
   !   ECO1    ROP0095         12-Jun-1986
   !           Check for nodes in a cluster when
   !           running as a server.

41) MBXDRIVER (patch image)

   ! MBXDRIVER.EXE
   !
   !   ECO01   LJK4027         07-May-1986
   !           MODULE: MBXDRIVER
   !           Pick up address of associated logical name block under the
   !           protection of the logical name mutex. This change corresponds
   !           to similar changes to DISMOUNT and MBDRIVER in the SYS
   !           facility.
```

```
42) MONITOR (patch image)

   ! MONITOR.EXE
   !
   !   ECO001                  28-Apr-1986
   !           MODULE: COLLECTION_EVENT/CLASS_INIT
   !           During class initialization, check to
   !           determine whether we are monitoring a multiprocessor.
   !           If we are, larger data buffers are needed for the
   !           modes class. This code path was being taken more than
   !           once if more than one dual processor was being monitored.
   !           This resulted in a divide-by-zero error with the CLUSTER
   !           class and bogus data for MODES and SYSTEM classes.

43) MP (new image)

   ! MP.EXE
   !
   !           SJF             25-Jun-1986
   !           MODULE: MPLOAD
   !           Fix wrong register usage in MPS$UNLOAD
   !
   !           SJF             16-May-1986
   !           MODULES: MPAST, MPINT, MPLOAD, CMODSSDSP
   !           Correct bugs around timeout of attached processor,
   !           potential clearing of ASTACT bits in wrong PCB,
   !           and 8800 attached processor boot.

44) MP_8NN (miscellaneous fix)

   ! MP_8nn.MSKEXE
   !
   !           SJF             25-Jun-1986
   !           MODULE: MPLOAD
   !           Fix wrong register usage in MPS$UNLOAD.
   !
   !           SJF             16-May-1986
   !           MODULES: MPAST, MPINT, MPLOAD, CMODSSDSP
   !           Correct bugs around timeout of attached processor,
   !           potential clearing of ASTACT bits in wrong PCB,
   !           and 8800 attached processor boot.

45) MP_8SS (miscellaneous fix)

   ! MP_8SS.MSKEXE
   !
   !           SJF             25-Jun-1986
   !           MODULE: MPLOAD
   !           Fix wrong register usage in MPS$UNLOAD.
   !
   !           SJF             16-May-1986
   !           MODULE: MPAST,MPINT,MPLOAD,CMODSSDSP
   !           Correct bugs around timeout of attached processor,
   !           potential clearing of ASTACT bits in wrong PCB,
   !           and 8800 attached processor boot.

46) MTAAACP (patch image)

   ! MTAAACP.EXE
   !
   !   ECO01   JWT0241         16-May-1986
   !           MODULE: END_OF_VOL
   !           When encountering a serious exception, clear
   !           it before calling START_VIO.  Failure to clear
   !           the serious exception for HSC tapes was causing
   !           the process to hang and lock up the drive.
```

```
47) NCP (patch image)

    ! NCP.EXE
    !
    !   ECO001  SFN004          12-May-1986
    !           MODULE: NCPTABLES
    !           Modify the ASCII string associated with the line BUFFER SIZE
    !           parameter.  It used to read "UNA device buffer size"; change to
    !           read "Device buffer size".

48) NETACP (patch image)

    ! NETACP.EXE
    !
    !           *** NOTE: ECOs 14 through 16 were from the V4.4 mandatory update.
    !           ECO 17 was assigned a number and we then decided to start
    !           renumbering from ECO 1.  Once all ECOs (i.e., up to and
    !           including 13) are used we must skip up to ECO 18.
    !
    !   ECO014  PRB014          27-Mar-1986
    !           MODULE: NETACPTRN
    !           Work around problem of misaligned LTB structure.
    !           For now, save size of structure when allocated, and restore it
    !           just before it's deallocated. Also fix type field.
    !
    !   ECO015  PRB015          27-Mar-1986
    !           MODULE: NETACPTRN
    !           Copy alias link region to new LTB when enlarging LTB on the fly.
    !
    !   ECO016  PRB016          01-Apr-1986 (ignore implications)
    !           MODULE: NETCLUSTR
    !           Dequeue lock to a node's ILR when dequeuing its IDL. Otherwise,
    !           as nodes come and go in the cluster, NETACP's ENQLM is slowly
    !           eroded.
    !
    !   ECO017  BAS             11-Apr-1986
    !           MODULE: NETCNFDLL
    !           Change the DEVTRN entry for DSV to allow multiple units on
    !           a controller.
    !
    !   ECO01   LY              30-Apr-1986
    !           MODULE: NETTRN - code
    !           NETTRN - NET_IMPURE psect
    !           Change timer handling to use delta time instead of absolute.
    !
    !   ECO02   PRB             08-May-1986
    !           MODULE: NETCONFIG
    !           Change version number for V4.5.
    !
    !   ECO03   PRB             08-May-1986
    !           MODULE: NETCNFDLL
    !           Fix problem where NETACP would remember the volatile setting of
    !           the PSI parameter MICROCODE DUMP from one call to the next.
    !
    !   ECO04   PRB             09-May-1986
    !           MODULE: NETDLLTRN
    !           Don't allocate cost/hops buffer in end node on designated router
    !           transition.
    !
    !   ECO05   SN              14-May-1986
    !           Fix the DSV device table entry in module NETCNFDLL.
    !
```

```
!   ECO06   MMD394          16-May-1986
!           Fix the NET$PROC_XWB such that if an error occurred while
!           processing the fields in the CI message the logical link
!           is not created and the XWB is deleted.
!
!   ECO07   SN              16-May-1986
!           Add algorithm into NETACP such that NETACP will not start up
!           MOM unless 1) the requesting node is in the database, 2) the
!           MOP message has a software ID in it, 3) the request is not for
!           the multicast address.
!
!   ECO08   SN              16-May-1986
!           Add new formatting routine to NETEVTLOG to account for the
!           event to be output if NETACP chooses not to start up MOM.
!
!   ECO09   PRB             01-Jun-1986
!           MODULE: NETACPTRN
!           Fixes problem with LTB getting corrupted when number of local links
!           is decreased while number of alias links is increased.
!
!   ECO10   PRB             14-Jul-1986
!           ** reserved **
!
!   ECO11   PRB             23-Jul-1986
!           ** reserved **
!
!   ECO12   LMP0384         06-Aug-1986
!           MODULE: NETCNFDLL.MAR
!           Ensure that the Ethernet object's UCB is appropriately protected.

49) NETCONFIG (edit text file)

! NETCONFIG.COM
!
!   ECO01   TRC0041         26-Mar-1986
!           Re-add the line for PA that got dropped in X-5.

50) NETDRIVER (patch image)

! NETDRIVER.EXE
!
!   ECO03   PRB03           27-Mar-1986
!           MODULES: NETDRVNSP.MAR, NETDRVXPT.MAR
!           If packet is marked "return to sender", use the source link ID,
!           not the destination link ID, in selecting which node in the
!           cluster should get the packet. This prevents unwarranted link
!           timeouts (instead of timely notification) when attempting to
!           access an unreachable node with an object with
!           OBJECT ALIAS OUTGOING enabled, which results if the returned
!           packet is misdirected within the cluster.

51) NMLSHR (patch image)

! NMLSHR.EXE
!
!   ECO01   BAS             10-Apr-1986
!           MODULE: NMLSEDEST
!           Correct database IDs for six destination parameters of
!           the X29-SERVER database: SDTE, RDTE, CDTE, IDTE, RED, and NET.
```

```
52) NODRIVER (patch image)

   !NODRIVER.EXE
   !
   !   ECO03   MMD0381         08-Apr-1986
   !           Fix to the DDCMP's REQUEUE_XMT routine to save R2
   !           around the call to DDCMP$SETCRC.

53) PADRIVER (new image)

   !  PADRIVER.EXE
   !
   !   ECO01   EJL0001         23-Apr-1986
   !           MODULE: PADRIVER
   !           Print warnings for CI microcode rev 2 through 5.
   !           Prevent a system crash when no microcode is available.
   !           Correct alignment of the buffer descriptor table.
   !           Correct port timeout debug check.
   !           Correct pool deallocation for unrecognized packets.
   !           Correct errors in new device support.

54) PATCH (patch image)

   ! PATCH.EXE
   !
   !   ECO01   JAK0001         16-May-1986
   !           JMS0001         02-Apr-1986
   !           MODULE: PATWRT
   !           Fix logic which computes the next block address for multiblock
   !           image headers when updating the base VBN in the ISDs.

55) PBDRIVER (new image)

   ! PBDRIVER.EXE
   !
   !   ECO01   WCY0050         01-May-1986
   !           MODULE: PBDRIVER
   !           Change the BDT alignment from quadword to octaword
   !           in FPC$INITIAL routine.

56) PUDRIVER (new image)

   ! PUDRIVER.EXE
   !
   !   ECO01   RLRPURHANG      14-May-1986
   !           MODULE: PUDRIVER
   !           Software workarounds for two problems that sometimes
   !           cause UDA ports to hang.
   !
   !           1. Introduce ability to alleviate "lost interrupts"
   !           by calling POLL_RSPRING from PU$SA_POLL under appropriate
   !           conditions.
   !
   !           2. In ISR (PU$INT) copy PDT$W_CMDINT and PDT$W_RSPINT to
   !           UCB$W_PU_CMDINT and UCB$W_PU_RSPINT respectively and only
   !           test for PURGE requests if both these flags are zero.
   !           This corrects a timing error in UDA50 that sometimes leaves
   !           UDA50 hanging waiting for a signal that purge has been done.
   !
   !           RLRGLBPAGE      27-Aug-1986
   !           Fix problem encountered when doing unaligned I/O
   !           to or from a global page.  Fix is in SETUP_COPY_SEGx
   !           routines.
```

```
57) REMACP (patch image)

   ! REMACP.EXE
   !
   !   EC001   DSS0001         17-Mar-1986
   !           MODULE: REMPROTCL
   !           Set the UCB for a remote device online only after
   !           the address of CTDRIVER has been inserted.

58) RMS (new image)

   ! RMS.EXE
   !
   !   EC01    JEJ0301         01-Apr-1986
   !           MODULE: RMOSTALL
   !           Check for failure in the $DCLAST call in order
   !           to avoid hanging a number of processes waiting
   !           for a lock.
   !
   !           PJH0001         07-Apr-1986
   !           MODULE: RM1GETINT
   !           Timeouts on record locks for shared sequential file
   !           can cause process to be deleted. Problem occurs when wait-on-
   !           timeout must give up bucket lock and doesn't go back and
   !           get bucket lock after record lock is granted. Patch is to
   !           go back and get bucket lock.
   !
   !           KPS031          09-Apr-1986
   !           MODULE: RMSORNDWN
   !           Ensure that rundown will check to see if a rundown
   !           is already active before proceeding. The
   !           problem was that batch jobs in rundown that were deleted with
   !           a DELETE/QUEUE/ENTRY command would sometimes incur an executive mode
   !           bugcheck because two rundown threads were active at the same
   !           time.
   !
   !           KPS032          09-Apr-1986
   !           MODULE: RMSOSETDD
   !           Ensure that SYS$SETDDIR will set and clear the
   !           RMS active bit around operations in its internal parse routines
   !           that allocate RMS data structures. This prevents an executive mode
   !           bugcheck that was possible if SETDDIR was active when a call
   !           was made to RMS rundown. This most commonly would occur in
   !           batch jobs that were deleted by the job controller sending an
   !           executive mode AST to the target process.
   !
   !           JWT0236         29-Apr-1986
   !           Use different key buffer for scratch area.  RMS was
   !           using key buffer 5 and destroying the value there.
   !
   !           JWT0237         29-Apr-1986
   !           Fix "$UPDATE Corrupter."  RMS does a scan to
   !           determine the optimal split point.  As is the usual
   !           practice, RMS saves a little bit of state at each
   !           potential split point and goes on to evaluate the
   !           next potential split point.  Once the current split
   !           point is evaluated as less optimal than the
   !           previous, things can only go downhill.  So RMS restores
   !           the state from the previously evaluated split point
   !           and splits there.
   !
```

```
!           RMS was not restoring all of the saved state info
!           from the previous split point, however.  Specifically,
!           SAVE_REC_W_LO, a flag that indicates whether or not
!           the new record ends up in the new (hi) or old (lo)
!           bucket was not being restored.  This was leaving
!           strange configurations of index records and
!           continuation buckets when lots of duplicates were
!           present.
!
!           PJH002          29-Apr-1986
!           Fix deadlock in the RMS last chance handler when a process
!           is run down as a result of a STOP/ID on the process. The
!           deadlock is possible only on a process which has a file
!           using global buffers opened by more than one FAB within
!           the process.
!
!           JEJ0294         18-Mar-1986
!           MODULE: RMOSETDID
!           RMS was incorrectly handling full 16-directory-deep
!           requests.  This was due to a directory path cache that
!           was one element too short.  The directory path cache pruning
!           code went berserk when it found this.  Correct it by
!           allocating a larger directory path cache.
!
!           JEJ0310         09-May-1986
!           MODULE: RMOBUFMGR
!           Further optimize the disk buffer clearing logic in
!           order to speed up $CONNECT.
!
!           JEJ0311         11-May-1986
!           MODULE: NTOCREATE
!           Properly propagate UPI sharing option to RSX-based
!           remote partners.
!
!           JEJ0312         12-May-1986
!           MODULE: NTOLNKCSH
!           Add support for timing out cached logical links.  If the
!           link is not reused within 30 seconds of being cached, it
!           is deaccessed and deassigned.
!
!   EC02    JEJ0340         14-Jul-1986
!           MODULE: RM1GET
!           Bucket locks may not be returned if a $GET encounters
!           end-of-file at the wrong moment.
!
!           JEJ0342         21-Jul-1986
!           MODULE: RMSOREWIN
!           $REWIND incorrectly dropped shared sequential files out
!           of "connected for append" mode, thus causing later
!           append operations to potentially corrupt the data file.
!
!           JEJ0344         22-Jul-1986
!           MODULE: RMOCACHE
!           An $EXTEND operation may incorrectly fail on a shared
!           sequential file if VBN 1 had been previously cached with
!           deferred writeback enabled.
!
!           JEJ0347         29-Jul-1986
!           MODULE: RM1GETINT
!           Setting USZ to zero would fail to reset the BDB pointer
!           before exiting with an error.
```

```
59) RUNDET (patch image)

    ! RUNDET.EXE
    !
    !   ECO1    ROP0094         15-May-1986
    !           Correct LIB$CVT_DTIME call to correctly
    !           deal with delta times greater than 24
    !           hours.

60) SCNRTL (patch image)

    ! SCNRTL.EXE
    !
    !   ECO01   MIZ             05-May-1986
    !           MODULE: SCN$START
    !           Use proper length for string input stream descriptor.
    !
    !   ECO02   MIZ             05-May-1986
    !           MODULES: SCN$UTIL, SCN$DBGEXT, SCN$ERROR
    !           Return dynamic string, if requested, from SCN$GET_TOKEN_NAME.
    !           Initialize output descriptor in calls to SCN$GET_TOKEN_NAME.
    !
    !   ECO03   MIZ             05-May-1986
    !           MODULE: SCN$PRUNE
    !           Fix bug in pruning algorithm for interior tree nodes.

61) SDA (patch image)

    ! SDA.EXE
    !
    !   ECO01   SJF4501         16-May-1986
    !           MODULE: CRASH
    !           Correct error message for new members of 8NN CPU family.

62) SETPO (patch image)

    ! SETPO.EXE
    !
    !   ECO01   ACG0516         28-Mar-1986
    !           MODULE: SETPASSWORD
    !           Correct a problem that could cause occasional access
    !           violation faults in the processing of SET PASSWORD
    !           commands.
    !
    !   ECO02   LMP0366         01-Aug-1986
    !           MODULE: SETPWD
    !           Correct severity of "no privilege" message in
    !           unsuccessful attempts to set the system password.

63) SETRIGHTS (new image)

    ! SETRIGHTS.EXE
    !
    !   ECO01                   11-Jun-1986
    !           Provide the SETRIGHTS image that was missing from the base
    !           MicroVMS system.  This eliminates the noise message that appears
    !           when the system is booted.
```

```
64) SHWCLSTR (new image)

   ! SHWCLSTR.EXE
   !
   !   ECO01   PRD0214         01-May-1986
   !           MODULE: SHWCLSTR.MAR
   !           Display remote-port number for CI only.
   !           Remove dependencies on hardware-type table.
   !           Allow unlimited removal of systems by
   !           node name, ID or hardware type.
   !           Properly restore default classes after INIT.
   !           Refresh after ADD or REMOVE field error.
   !           Restore scrolling region properly if initial
   !           display exceeds terminal length.

65) SMBSRVSHR (new image)

   ! SMBSRVSHR.EXE
   !
   !   ECO0001 RRB0001         07-May-1986
   !           Fix problem with files left open.  Specifically, device control
   !           library modules, main input files.  Fixes yet another instance of
   !           unexpected symbiont process termination.

66) SMGRTL (miscellaneous fix)

   ! SMGDEF.SDI
   !
   !   1-002   TS              01-May-1986
   !           MODULE: $SMGDEF
   !           Provide $SMGDEF as one module instead
   !           of two appended modules.

67) STABACKIT (edit text file)

   ! STABACKIT.COM
   !
   !   ECO01   CWH4005         16-Apr-1986
   !           Make sure that WRITEBOOT is on the system before
   !           using it, because MicroVMS has VMB but not WRITEBOOT.

68) STABACKIT2 (edit text file)

   ! STABACKIT.COM
   !
   !   ECOnn   JJ00022         27-Feb-1986
   !           Provide hooks to build RX50 kits which can be removed
   !           when booting standalone BACKUP. Track the SYSBOOT change
   !           which removes the need to use B/20000 when booting from
   !           TK50s.

69) STABACKUP (patch image)

   ! STABACKUP.EXE
   !
   !   ECO007  KGW00022        30-May-1986
   !           MODULE: TAPEUTIL
   !           Patch MUST be applied to VMS V4.4 systems ONLY.
   !
   !           Excessive parity errors on a TA78 would result in
   !           access to the tape being lost.
   !           Repair this by unloading the tape if the RESTART
   !           option is taken from the error handler.
```

```
70) STARLET (miscellaneous fix)

   ! LIB.MLB
   !
   !   EC001   JLV001          13-May-1986
   !           MODULE: $MSGDEF
   !           Add symbols for VAX Workstation VT220 emulator
   !           mailbox messages.
   !
   !   EC002   MMD             15-May-1986
   !           MODULE: $DCDEF
   !           Add new device type symbol.
   !
   !   EC003   CBD             19-May-1986
   !           MODULE: $PRDEF
   !           Add 8550/8700 processor symbols

71) SUBMIT (patch image)

   ! SUBMIT.EXE
   !
   !   EC001   JES0001         16-Apr-1986
   !           MODULE: SUBMIT
   !           Fix the /NOKEEP qualifier.
   !           Change call to parse /KEEP qualifier from PARSE_IF_TRUE to
   !           PARSE_IF_TRUE_FALSE so that the presence of a /NOKEEP qualifier
   !           would get sent to the job controller.

72) SYS (patch image)

   ! SYS.EXE
   !
   !   EC097   JAY             02-Jun-1986
   !           MODULES: MDAT, SYSPARAM
   !           Increase the size of the nonpaged read-only patch area
   !           by 512 bytes to accommodate large patches.
   !
   !   EC060   DWT82638        24-Feb-1986
   !           MODULE: SYSGETLKI
   !           Correct handling of buffer overflow for $GETLKI item
   !           LKI$_LOCKS for a locally mastered lock.
   !
   !   EC063   DAS/PMV         21-Jan-1986
   !           MODULE: CMODSSDSP
   !           Change RUF service vectors to do CHMEs instead of CHMKs
   !           so that RUF will run in executive mode.  Also change the register
   !           save mask to also save registers R7 through R11.
   !
   !   EC064   CJM/TBE         18-Jul-1986
   !           MODULE: INIT, SYSCOMMON
   !           Set version to X4.5.
   !
   !   EC065   LJK4026         21-Apr-1986
   !           MODULE: SYSIMGACT
   !           Use correctly calculated access mode when deleting address
   !           space mapped as part of a failed activation.
   !
   !   EC066   JJW/JAY         14-May-1986
   !           MODULE: INIT
   !           Check for the existence of a subport driver.  If one exists,
   !           create the appropriate data structures in the I/O database.
   !
```

```
!           CWH0045         16-May-1986
!           MODULE: SYSGETDVI
!           Use spare item code to return MSCP unit number.
!
!           CWH0045.1       04-Jun-1986
!           MODULE: SYSGETLKI
!           Fix synchronization problem in $GETLKI.
!
!           WES0045         04-Jun-1986
!           MODULE: SYSENQDEQ
!           Clear R9 in DEQALL loop to avoid value block corruption under
!           obscure circumstances.
!
!   EC067   CJM             03-Sep-1986
!           MODULE: INIT, SYSCOMMON
!           Set version to V4.5.
!
!   EC068   LJK4027         05-May-1986
!           MODULES: DISMOUNT, MBDRIVER
!           The check for an associated name must be synchronized with
!           possible parallel explicit deletion of the logical name.
!
!   EC069   ACG0520         16-May-1986
!           MODULE: SYSCRMPSC
!           Correct a stack overflow problem in the auditing of
!           accesses to global sections that can cause a
!           "kernel stack not valid" bugcheck.
!
!   EC070   CBD0015         16-May-1986
!           Change NOTRULUCB bugcheck to FATMEMERR.
!
!   EC071   WMC0001         08-Jul-1986
!           Fix to ECO 30 to transfer data in correct direction.
!
!   EC072   LJK4029         10-Jun-1986
!           MODULE: SYSQIOFDT (really it's ECO 62)
!           Allow a byte count of zero to pass limit checks in
!           EXE$READCHKR and EXE$WRITECHKR. Only disallow very large
!           byte counts (with bit 31 set).
!
!           LJK4030         21-Aug-1986
!           MODULE: SYSLNM
!           Effectively eliminate pointer from logical name block to UCB
!           or MTL (by changing its index from -127 to -125). When
!           coupled with ECO 55, this completely decouples the logical
!           name structures from either a UCB or a MTL, preventing
!           spurious channel assignments.
!
!   EC073   TCM0001         22-Aug-1986
!           MODULE: SYSUPDSEC
!           Fix $UPDSEC to fault in page table pages when updating a
!           global section.
!
!           CWH4004         25-Aug-1986
!           Increase item JPI$_PROC_INDEX to longword from byte.
```

```
73) SYSBOOT (new image)

   ! SYSBOOT.EXE
   !
   !   ECO01   JJ00021         10-Feb-1986
   !           Add hooks that allow the RX50 to be removed when
   !           booting standalone BACKUP. Also remove the need
   !           to use B/20000 for booting TK50 standalone BACKUP.
   !
   !   ECO02   JAY0001         12-May-1986
   !           Add support to process EXE$GL_AUXDRLIST in
   !           BOOTDRIVR.
   !
   !   ECO03   CBD0012         16-May-1986
   !           Add 8550/8700 name support.

74) SYSGEN (new image)

   ! SYSGEN.EXE
   !
   !   ECO01   EMB0001         16-Apr-1986
   !           MODULES: AUTOCONFG, LOADER, SHOWADAP
   !           Minor bug fixes for VAXBI devices.
   !
   !   ECO02   WCY0052         14-May-1986
   !           MODULE: AUTOCONFG
   !           Add device support for BVPSSP ports in systems
   !           also containing UQSSP ports.
   !
   !   ECO03   EMB0219         15-May-1986
   !           MODULES: AUTOCONFG, CONFIG
   !           Correctly configure LDP devices and add a device entry
   !           to the UNIBUS device list.
   !
   !   ECO04   WCY0056         16-May-1986
   !           MODULE: AUTOCONFG
   !           Correct bug introduced in WCY0051.

75) SYSLOA730 (patch image)

   ! SYSLOA730.EXE
   !
   !   ECO01   ROW0562         15-May-1986
   !           MODULE: MOUNTVER
   !           Fix VALIDATE_HOME to handle clusterwide inconsistencies
   !           created by SET VOLUME/LABEL.  Most of the code to handle this
   !           problem already exists.  However, VALIDATE_HOME renders it
   !           useless because it improperly returns an error status.

76) SYSLOA750 (patch image)

   ! SYSLOA750.EXE
   !
   !   ECO01   ROW0562         15-May-1986
   !           MODULE: MOUNTVER
   !           Fix VALIDATE_HOME to handle clusterwide inconsistencies
   !           created by SET VOLUME/LABEL.  Most of the code to handle this
   !           problem already exists.  However, VALIDATE_HOME renders it
   !           useless because it improperly returns an error status.
```

```
77) SYSLOA780 (patch image)

    ! SYSLOA780.EXE
    !
    !   ECO01   ROW0562         15-May-1986
    !           MODULE: MOUNTVER
    !           Fix VALIDATE_HOME to handle clusterwide inconsistencies
    !           created by SET VOLUME/LABEL.  Most of the code to handle this
    !           problem already exists.  However, VALIDATE_HOME renders it
    !           useless because it improperly returns an error status.

78) SYSLOA790 (patch image)

    ! SYSLOA790.EXE
    !
    !   ECO01   ROW0562         15-May-1986
    !           MODULE: MOUNTVER
    !           Fix VALIDATE_HOME to handle clusterwide inconsistencies
    !           created by SET VOLUME/LABEL.  Most of the code to handle this
    !           problem already exists.  However, VALIDATE_HOME renders it
    !           useless because it improperly returns an error status.

79) SYSLOA8NN (new image)

    ! SYSLOA8NN.EXE
    !
    !   ECO01   CBD0027         03-Sep-1986
    !           MODULE: ADPERR8NN
    !           Don't crash on IMR errors. Ignore them for now.
    !
    !           EMB0246         31-Jul-1986
    !           MODULE: OPDRV8NN
    !           Change MOVZBL instruction to CVTBL so that
    !           the following BLSS instruction will work.
    !
    !           CBD0015         16-May-1986
    !           a. Issue FATMEMERR BUGCHECK on fatal memory errors,
    !              not a MACHINECHK (MCHECK8NN).
    !           b. Add 8550/8700 name support.
    !
    !           CBD0007         06-May-1986
    !           a. Typo in MCHK that mixed _V and _M bit specifiers in
    !              memory error checking.
    !           b. Add console message to MCHK when cache is turned
    !              off because of a high error rate.
    !
    !           EJL0053         09-Apr-1986
    !           Fix ADPSUB8NN to wait two milliseconds after initiating
    !           self test on the CIBCI.
    !
    !           MSH0237         01-Apr-1986
    !           Fix OPDRV8NN so that timeouts to the console transmitter
    !           port won't cause a system crash on the next attempt to
    !           transmit data.
```

```
80) SYSLOA8SS (new image)

   ! SYSLOA8SS.EXE
   !
   !           WCY0054         16-May-1986
   !           Add support for BVPSSP ports in CALC_CTRLLTR.
   !
   !           EMB0001         09-May-1986
   !           Correctly set up the hardware type in the system
   !           block (SB) to distinguish between an 8200 and an
   !           8300.
   !
   !           EJL0052         09-Apr-1986
   !           Fix ADPSUB to wait two milliseconds after initiating
   !           self test on the CIBCI.

81) SYSLOAUV1 (patch image)

   ! SYSLOAUV1.EXE
   !
   !   ECO01   ROW0562         15-May-1986
   !           MODULE: MOUNTVER
   !           Fix VALIDATE_HOME to handle clusterwide inconsistencies
   !           created by SET VOLUME/LABEL.  Most of the code to handle this
   !           problem already exists.  However, VALIDATE_HOME renders it
   !           useless because it improperly returns an error status.

82) SYSLOAUV2 (patch image)

   ! SYSLOAUV2.EXE
   !
   !   ECO01   ROW0562         15-May-1986
   !           MODULE: MOUNTVER
   !           Fix VALIDATE_HOME to handle clusterwide inconsistencies
   !           created by SET VOLUME/LABEL.  Most of the code to handle this
   !           problem already exists.  However, VALIDATE_HOME renders it
   !           useless because it improperly returns an error status.
   !
   !   ECO02   DGB0158         09-Jun-1986
   !           Work around VMB problem, where two bad pages of memory are
   !           reported on 16MB MicroVAX II systems.

83) SYSLOAWS1 (patch image)

   ! SYSLOAWS1.EXE
   !
   !   ECO01   ROW0562         15-May-1986
   !           MODULE: MOUNTVER
   !           Fix VALIDATE_HOME to handle clusterwide inconsistencies
   !           created by SET VOLUME/LABEL.  Most of the code to handle this
   !           problem already exists.  However, VALIDATE_HOME renders it
   !           useless because it improperly returns an error status.
```

```
84) SYSLOAWS2 (patch image)

    ! SYSLOAWS2.EXE
    !
    !   EC001   ROW0562         15-May-1986
    !           MODULE: MOUNTVER
    !           Fix VALIDATE_HOME to handle clusterwide inconsistencies
    !           created by SET VOLUME/LABEL.  Most of the code to handle this
    !           problem already exists.  However, VALIDATE_HOME renders it
    !           useless because it improperly returns an error status.
    !
    !   EC002   DGB0160         12-Jun-1986
    !           Work around VMB problem, where two bad pages of memory are
    !           reported on 16MB MicroVAX II systems.

85) SYSLOAWSD (patch image)

    ! SYSLOAWSD.EXE
    !
    !   EC001   ROW0562         15-May-1986
    !           MODULE: MOUNTVER
    !           Fix VALIDATE_HOME to handle clusterwide inconsistencies
    !           created by SET VOLUME/LABEL.  Most of the code to handle this
    !           problem already exists.  However, VALIDATE_HOME renders it
    !           useless because it improperly returns an error status.
    !
    !   EC002   DGB0161         12-Jun-1986
    !           Work around VMB problem, where two bad pages of memory are
    !           reported on 16MB MicroVAX II systems.

86) SYSMSG (patch image)

    ! SYSMSG.EXE
    !
    !   EC001   KC0001          16-Apr-1986
    !           MODULE: BASMSG
    !           Patch the text of various messages for use by
    !           BASIC V3.0 and BASRTL.
    !
    !   EC002   KC0002          16-May-1986
    !           MODULE: BASMSG
    !           More message text changes.
    !
    !   EC003   KC0003          21-May-1986
    !           MODULE: BASMSG
    !           More message text changes.
    !
    !   EC004   KC0004          16-Jun-1986
    !           MODULE: BASMSG
    !           Change text of FROLINOEG, add ILLCOLMIX.

87) TFDRIVER (new image)

    ! TFDRIVER.EXE
    !
    !   EC00001 BJT0020         25-Mar-1986
    !           SPR # 11-79345 - Handle error recovery resulting
    !           in read reverse.
```

```
88) TMDRIVER (new image)

   ! TMDRIVER.EXE
   !
   !   EC00001 BJT0032         25-Mar-1986
   !           SPR # 11-77416 - Fix bug which clears tape mark bit in SENSEMODE
   !           at 1600 bpi.
   !
   !           SPR # 11-81269 - Set end-of-tape bit in DEVDEPEND whenever
   !           end-of-tape is seen.

89) TPU$CCTSHR (new image)

   ! SYS$LIBRARY:TPU$CCTSHR
   !
   !           STL0001         14-May-1986
   !           Fix support of terminal widths.

90) TPUSHR (patch image)

   ! TPUSHR.EXE
   !
   !   EC001   BMT0001         02-Apr-1986
   !           MODULE: TPUSHR.EXE
   !           Change the protection mask specified for the mailboxes used
   !           to communicate with the TPU subprocesses to allow only owner
   !           access.
   !
   !   EC002   BMT002          10-Apr-1986
   !           MODULE: TPUSHR.EXE
   !           Add check to GET_INFO built-in to check for sufficient
   !           parameters.

91) TRACE (new image)

   ! TRACE.EXE
   !
   !   EC001                   14-May-1986
   !           MODULE: TBKDPC.B32
   !           This module contains a fix for handling BASIC statement
   !           and line numbers correctly in tracebacks.

92) TTDRIVER (new image)

   ! TTDRIVER.EXE
   !
   !   EC002   MIR0713
   !           Fix some workstation problems with CTRL/Y and CTRL/C ASTs
   !           and subprocesses.  Add several port control bits to disable
   !           modem and connect/disconnect for a particular line using
   !           a port control field.
```

```
93) TUDRIVER (new image)

   ! TUDRIVER.EXE
   !
   !   ECO02   MAS0065         27-May-1986
   !           Fix connection walking bug that leads to spurious host clears
   !           of HSCs following virtual circuit failures during failover.
   !           Correct other miscellaneous problems involving host and
   !           controller timeouts.  Relevant module audit versions are
   !           TUDRIVER (X-23) and DUTUSUBS (X-30).
   !
   !   ECO01   PRD0227         16-May-1986
   !           MODULE: TUDRIVER
   !           Add support for static dualporting.
   !           Recognize unit flags upon system boot.

94) UETCLIGOO (edit text file)

   ! UETCLIGOO.COM

95) UETINITOO (patch image)

   ! UETINITOO.EXE
   !
   !   ECO01   PEL0001         12-May-1986
   !           MODULE: UETINITOO
   !           Add support for new processors.

96) UETSUPDEV (miscellaneous fix)

   ! SYS$TEST:UETSUPDEV.DAT
   !
   !   ECO03   JLW0002         04-May-1986
   !           MODULE: SYS$TEST:UETSUPDEV.DAT
   !           Add support for the AIE.

97) UETUNASOO (patch image)

   ! UETUNASOO.EXE
   !
   !   ECO01   JLW0005         31-Mar-1986
   !           MODULE: UETUNASOO.MAR
   !           Fix test so it won't hang when run on a standalone
   !           Ethernet.  Changed internal loop message sizes to accommodate
   !           the DELUA.
   !
   !   ECO02   JLW0008         04-May-1986
   !           MODULE: UETUNASOO.MAR
   !           Add support for the AIE.  Fix timer problem causing
   !           "Error in error test" diagnostic.

98) UPDATE_CONSOLE (edit text file)

   ! UPDATE_CONSOLE.COM
   !
   !   ECO01   WES0021         15-May-1986
   !           Change SYS$DISK reference to be SYS$SCRATCH.

99) VAXCRTL (miscellaneous fix)

   ! VAXCRTL.OLB
   !
   !           Replace object modules to VAXCRTL.OLB corresponding to patches made
   !           to VAXCRTL.EXE.
   !
```

```
!   EC001   CHH0061         10-Apr-1986
!           MODULE: C$VAXCIO (061)
!           Miscalculating the end-of-file byte offset and block number.
!
!   EC002   CHH0023         28-Apr-1986
!           MODULE: C$$MAIN (023), SHELL$CLINT (002)
!           Internal buffer overflow can happen in the SHELL$GET_ARGV
!           routine when trying to get "argv" and "argc" under DEC/SHELL.
!
!   EC003   CHH0013         28-Apr-1986
!           MODULE: SHELL$FROM_VMS (013)
!           Add checks in SHELL$TRANSLATE_VMS routine and COPY_TOKEN
!           routine to avoid buffer overflow.
!
!   EC004   CHH0018         28-Apr-1986
!           MODULE: SHELL$TO_VMS (018)
!           Modify SHELL$TO_VMS routine to avoid buffer overflow and to
!           handle the foreign filespecs correctly.
!
!   EC005   CHH0031         13-May-1986
!           MODULE: C$$DOPRINT (031)
!           Fix for SPR 87940.  Modify C$$DOSCAN routine to match a
!           character string pattern correctly.
!
!   EC006   CHH0062         14-May-1986
!           MODULE: C$VAXCIO (062)
!           Fix for SPR 87940.  Modify READ_STREAM subroutine to avoid
!           overwriting an internal I/O buffer.
!
!   EC007   CHH0063         27-May-1986
!           MODULE: C$VAXCIO (063)
!           Fix for SPR 88853.  Modify the LSEEK function to return the
!           correct byte offset after a write operation for record file.
!
!   EC008   CHH0063         28-May-1986
!           MODULE: C$VAXCIO (063)
!           Fix in the CHDIR function. SYS$DISK is not a terminal logical
!           name.

100) VAXCRTL (patch image)

! VAXCRTL.EXE
!
!   EC001   CHH0061         10-Apr-1986
!           MODULE: C$VAXCIO (061)
!           Miscalculating the end-of-file byte offset and block number in
!           subroutine WRITE_OUTPUT.
!
!   EC002   CHH0023         28-Apr-1986
!           MODULE: SHELL$CLINT (002) and C$$MAIN (023)
!           Internal buffer overflow can happen in the SHELL$GET_ARGV
!           routine when trying to get "argv" and "argc" under DEC/SHELL.
!
!   EC003   CHH0013         01-May-1986
!           MODULE: SHELL$FROM_VMS (013)
!           Add checks in SHELL$TRANSLATE_VMS routine and COPY_TOKEN
!           routine to avoid buffer overflow.
!
```

```
    !   EC004   CHH0018         01-May-1986
    !           MODULE: SHELL$TO_VMS (018)
    !           Modify SHELL$TO_VMS routine to avoid buffer overflow and to
    !           handle the foreign filespecs correctly.
    !
    !   EC005   CHH0031         13-May-1986
    !           MODULE: C$$DOPRINT (031)
    !           Fix for SPR 87940.  Modify C$$DOSCAN routine to match a
    !           character string pattern correctly.
    !
    !   EC006   CHH0062         13-May-1986
    !           MODULE: C$VAXCIO (062)
    !           Fix for SPR 87940.  Modify READ_STREAM subroutine to avoid
    !           overwrite an internal buffer.
    !
    !   EC007   CHH0063         27-May-1986
    !           MODULE: C$VAXCIO (063)
    !           Fix for SPR 88853.  Modify the LSEEK function to return the
    !           correct byte offset after a write operation for record file.
    !
    !   EC008   CHH0063         28-May-1986
    !           MODULE: C$VAXCIO (063)
    !           Fix in the CHDIR function. SYS$DISK is not a terminal logical
    !           name.
    !
    !   EC009   CJN0064         07-Aug-1986
    !           MODULE: C$VAXCIO (064)
    !           Fix in WRITE_TRANSFER to check for zero-length transfer and to do
    !           nothing if such is found.
    !
    !   EC010   CJN0064         07-Aug-1986
    !           MODULE: C$VAXCIO (064)
    !           Fix in WRITE function to not flush buffer if device is a SHELL
    !           pipe.
    !
    !   EC011   CJN064          07-Aug-1986
    !           MODULE: C$VAXCIO (064)
    !           Fix in _FLSBUF_STREAM to remove optimization of no preload of next block
    !           if read access is not permitted.

101) VAXCRTLG (miscellaneous fix)

    !
    !  VAXCRTLG.OLB
    !
    !           Replace object modules to VAXCRTLG.OLB corresponding to patches made
    !           to VAXCRTLG.EXE.
    !
    !   EC001   CHH0002         10-Apr-1986
    !           MODULE: C$ECVT (002)
    !           Passing 0 to FCVT function fails with ACCVIO.
    !
    !   EC006   CHH0031         13-May-1986
    !           MODULE: C$$DOPRINT (031)
    !           Fix for SPR 87940.  Modify C$$DOSCAN routine to match a
    !           character string pattern correctly.
```

```
102) VAXCRTLG (patch image)

    ! VAXCRTLG.EXE
    !
    !   EC001   CHH0002         10-Apr-1986
    !           MODULE: C$ECVT (002)
    !           Passing 0 to FCVT function fails with ACCVIO.
    !
    !   EC002   CHH0061         10-Apr-1986
    !           MODULE: C$VAXCIO (061)
    !           Miscalculating the EOF byte offset and block number in
    !           subroutine WRITE_OUTPUT.
    !
    !   EC003   CHH023          28-Apr-1986
    !           MODULE: SHELL$CLINT (002) and C$$MAIN (023)
    !           Internal buffer overflow can happen in the SHELL$GET_ARGV
    !           routine when trying to get "argv" and "argc" under DEC/SHELL.
    !
    !   EC004   CHH0013         01-May-1986
    !           MODULE: SHELL$FROM_VMS (013)
    !           Add checks in SHELL$TRANSLATE_VMS routine and COPY_TOKEN
    !           routine to avoid buffer overflow.
    !
    !   EC005   CHH0018         01-May-1986
    !           MODULE: SHELL$TO_VMS (018)
    !           Modify SHELL$TO_VMS routine to avoid buffer overflow and to
    !           correctly handle the foreign filespecs.
    !
    !   EC006   CHH0031         13-May-1986
    !           MODULE: C$$DOPRINT (031)
    !           Fix for SPR 87940.  Modify C$$DOSCAN routine to match a
    !           character string pattern correctly.
    !
    !   EC007   CHH0062         13-May-1986
    !           MODULE: C$VAXCIO (062)
    !           Fix for SPR 87940.  Modify READ_STREAM subroutine to avoid
    !           overwriting an internal buffer.
    !
    !   EC008   CHH0063         27-May-1986
    !           MODULE: C$VAXCIO (063)
    !           Fix for SPR 88853.  Modify the LSEEK function to return the
    !           correct byte offset after a write operation for record file.
    !
    !   EC009   CHH0063         28-May-1986
    !           MODULE: C$VAXCIO (063)
    !           Fix in the CHDIR function. SYS$DISK is not a terminal logical
    !           name.
    !
    !   EC010   CJN0064         07-Aug-1986
    !           MODULE: C$VAXCIO (064)
    !           Fix in WRITE_TRANSFER to check for zero length transfer and to do
    !           nothing if such is found.
    !
    !   EC011   CJN0064         07-Aug-1986
    !           MODULE: C$VAXCIO (064)
    !           Fix in WRITE function to not flush buffer if device is a SHELL
    !           pipe.
    !
    !   EC012   CJN064          07-Aug-1986
    !           MODULE: C$VAXCIO (064)
    !           Fix in _FLSBUF_STREAM, to remove optimization of no preload of next block
    !           if read access is not permitted.
```

```
103) VMB (new image)

    !  VMB.EXE
    !
    !   EC002   CBD0010         05-May-1986
    !           MODULE: VMB
    !           Output a message when greater than 10 percent of main
    !           memory tests bad.
    !
    !   EC001   EJL0002         23-Apr-1986
    !           MODULE: PABTDRIVR
    !           Correct errors in new device support.

104) VMSINSTAL (edit text file)

    ! VMSINSTAL.COM
    !
    !   EC001   JES0001         21-Apr-1986
    !           Fix problem with PROVIDE_FILE option C.  Option C
    !           ignored if system disk not identical to target disk.
    !
    !           Fix bug with printing of release notes in callback
    !           RELEASE_NOTES.
    !
    !           Add /PAGE to TYPE command in RELEASE_NOTES callback.
    !
    !           Fix bug in PROVIDE_IMAGE, where we were not checking
    !           status after a VMI$FIND call.
    !
    !           Add code for low-end cluster layered product the ability
    !           to install in the specific root.
    !
```

B Generic VAXBI Device Support in VAX/VMS

This document provides information needed to write and load a device driver for a non-DIGITAL-supplied device attached to the VAXBI bus. VAX/VMS Version 4.5 provides special support for such devices in the system initialization routines for the VAX 8200, VAX 8300, VAX 8500, VAX 8550, VAX 8700, and VAX 8800 systems. Because of the many and varied implementations of VAXBI devices, however, VAX/VMS support must of necessity be very general. Some devices may more fully utilize the VAXBI interface than others; a device may incorporate its interface initialization logic in microcode, whereas another may defer initialization to code in its driver.

The *VAXBI Options Handbook* includes a description and guidelines for possible VAXBI device implementations. Please refer to that manual for further discussion of all VAXBI topics discussed in brief in Section B.2 and elsewhere in this document.

B.1 Overview

A VAXBI device driver refers to the same data structures and contains the same routines as a traditional VAX/VMS driver. As this document presents only information specific to writing a driver for a non-DIGITAL-supplied device, it presumes an understanding of the material discussed in the *Writing a Device Driver for VAX/VMS* manual.

A VAXBI device driver deviates from the traditional VAX/VMS driver almost exclusively in code that initializes the VAXBI interface or supports direct-memory-access (DMA) transfers for devices that address memory across the VAXBI bus. Section B.4 discusses tasks that drivers of various VAXBI devices may perform in their initialization routines to supplement VAX/VMS initialization and that initialization performed by device microcode. Section B.5 contains a general discussion of how some VAXBI devices and their drivers manage DMA transactions.

Section B.3 describes those data structures the VAX/VMS adapter initialization routine creates and prepares for a generic VAXBI device, while Section B.7 discusses the method by which its driver can be loaded into the operating system. The final section of this document provides reference material and includes a description of the backplane interconnect interface chip (BIIC) registers and a summary of the IOC$ALLOSPT system routine.

B.2 VAXBI Concepts

The *VAXBI* serves as the I/O bus for the VAX 8200, VAX 8300, VAX 8500, VAX 8550, VAX 8700 and VAX 8800 systems (see Figure B–1).[1] Each of the VAX 8200 and VAX 8300 systems can have a single VAXBI; the VAX 8500, VAX 8550, VAX 8700, and VAX 8800 systems can have multiple VAXBI buses (see Figure B–2).

[1] The VAXBI is also the system bus for the VAX 8200 and VAX 8300 systems.

Figure B–1 VAX 8200 and VAX 8300 Systems

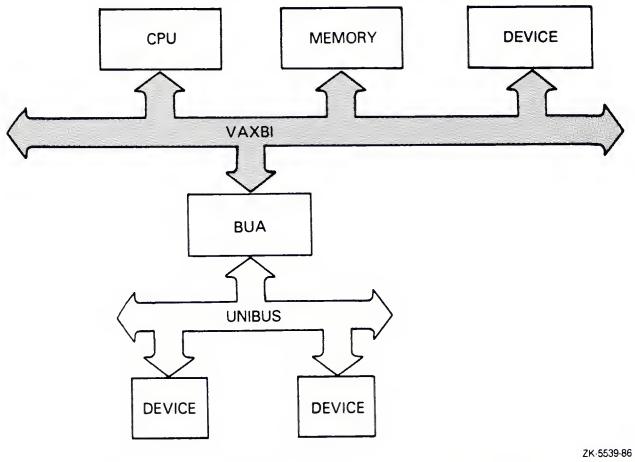
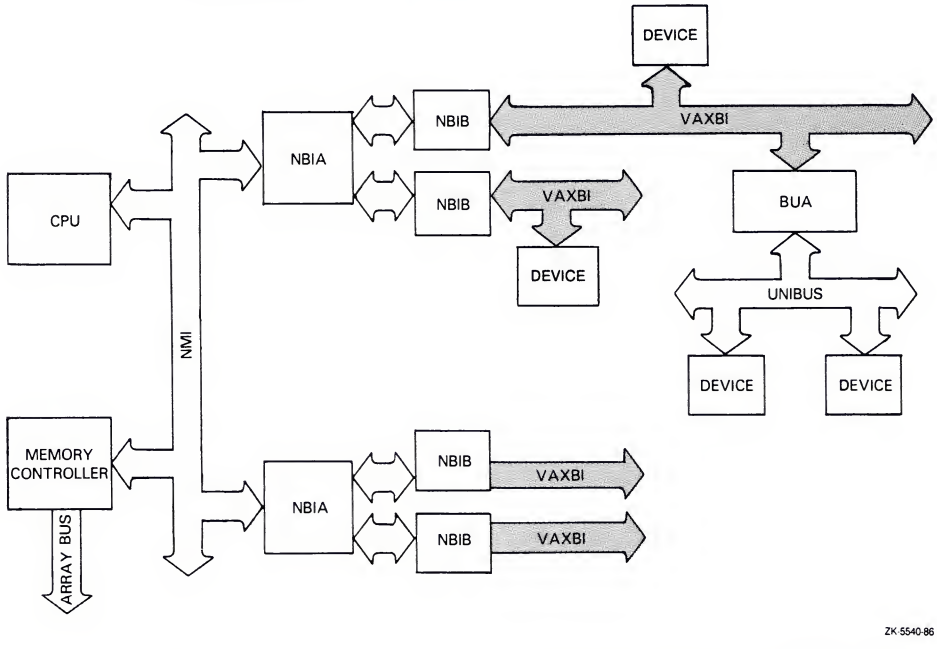


Figure B–2 VAX 8500, VAX 8550, VAX 8700, and VAX 8800 Systems



Each location on a VAXBI bus is called a *node*. A single VAXBI bus can service 16 nodes. In the case of the VAX 8200 and VAX 8300 systems, these nodes can be processors, memory, and adapters; the VAX 8500, VAX 8550, VAX 8700 and VAX 8800 systems permit only adapters to be attached to the VAXBI bus.[2] A node receives its *node ID*, a number from 0 to 15, from a plug on the VAXBI backplane slot into which the node module is inserted.

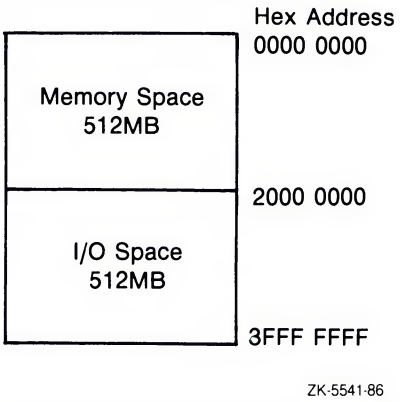
[2] For the VAX 8500, VAX 8550, VAX 8700, and VAX 8800, the NMI-to-BI adapter (NBIA/NBIB) or, more specifically, the NBIB, resides at a node on a VAXBI bus, monitoring and controlling transactions to the memory interconnect (NMI) where the processors and memory reside.

An *adapter* is a node that connects other buses, communication lines, and peripheral devices to the VAXBI bus. This document uses the term *device* to refer to a device or combination of devices serviced by a single adapter or controller.

B.2.1 VAXBI Address Space

Each VAXBI bus supports 30-bit addressing capability. This gigabyte of physical address space is split equally between memory and I/O space, as shown in Figure B–3.

Figure B–3 VAXBI Address Space



All memory locations on a VAXBI bus are addressed using physical addresses in VAXBI memory space (from 00000000_{16} through $1FFFFFFF_{16}$). A VAXBI device that accesses memory directly, or its driver, must perform virtual-to-physical translation before transmitting a memory address on the bus (see Section B.5).

VAXBI I/O space (physical addresses 20000000_{16} through $3FFFFFFF_{16}$) is partitioned as illustrated in Figure B–4. Figure B–5 shows the structure of an I/O-space address.

As shown in Figures B–4 and B–5, VAXBI architecture grants each of the 16 nodes on a VAXBI bus two discrete sections in I/O space.

Figure B–4 Description of VAXBI I/O Space

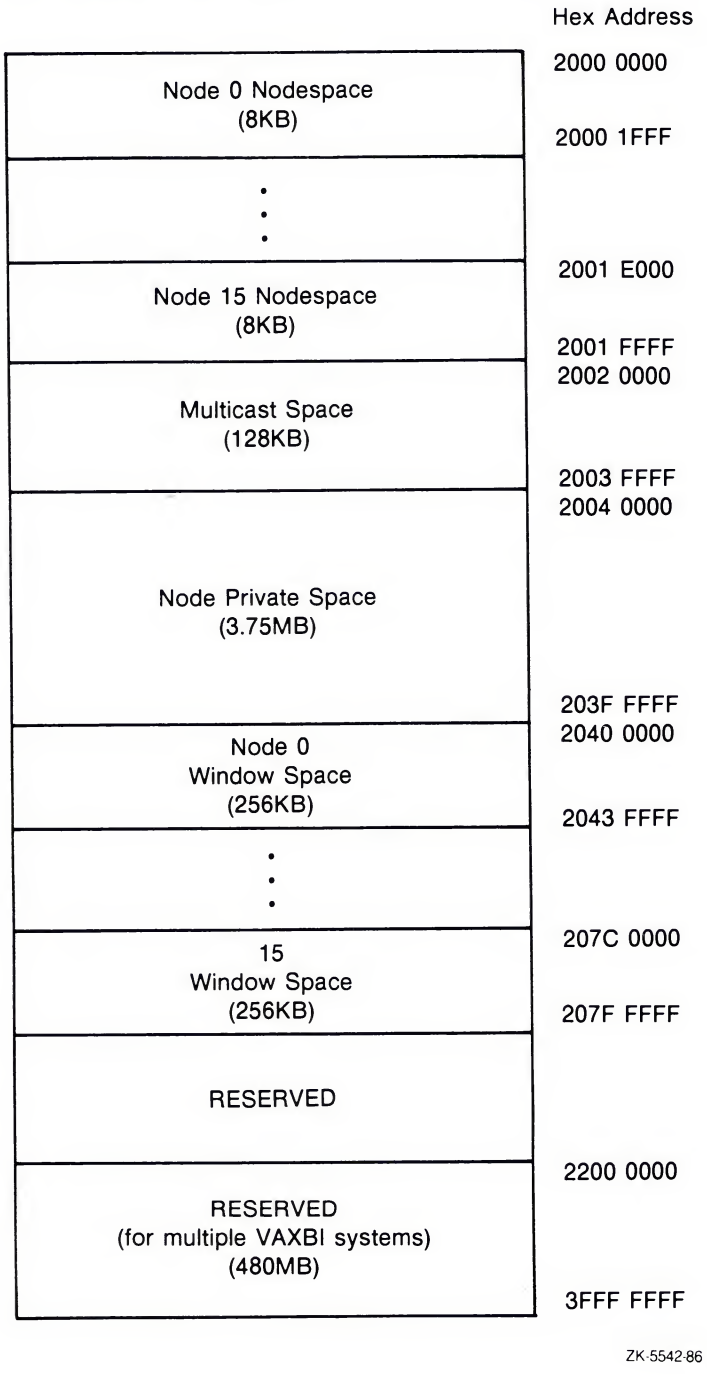
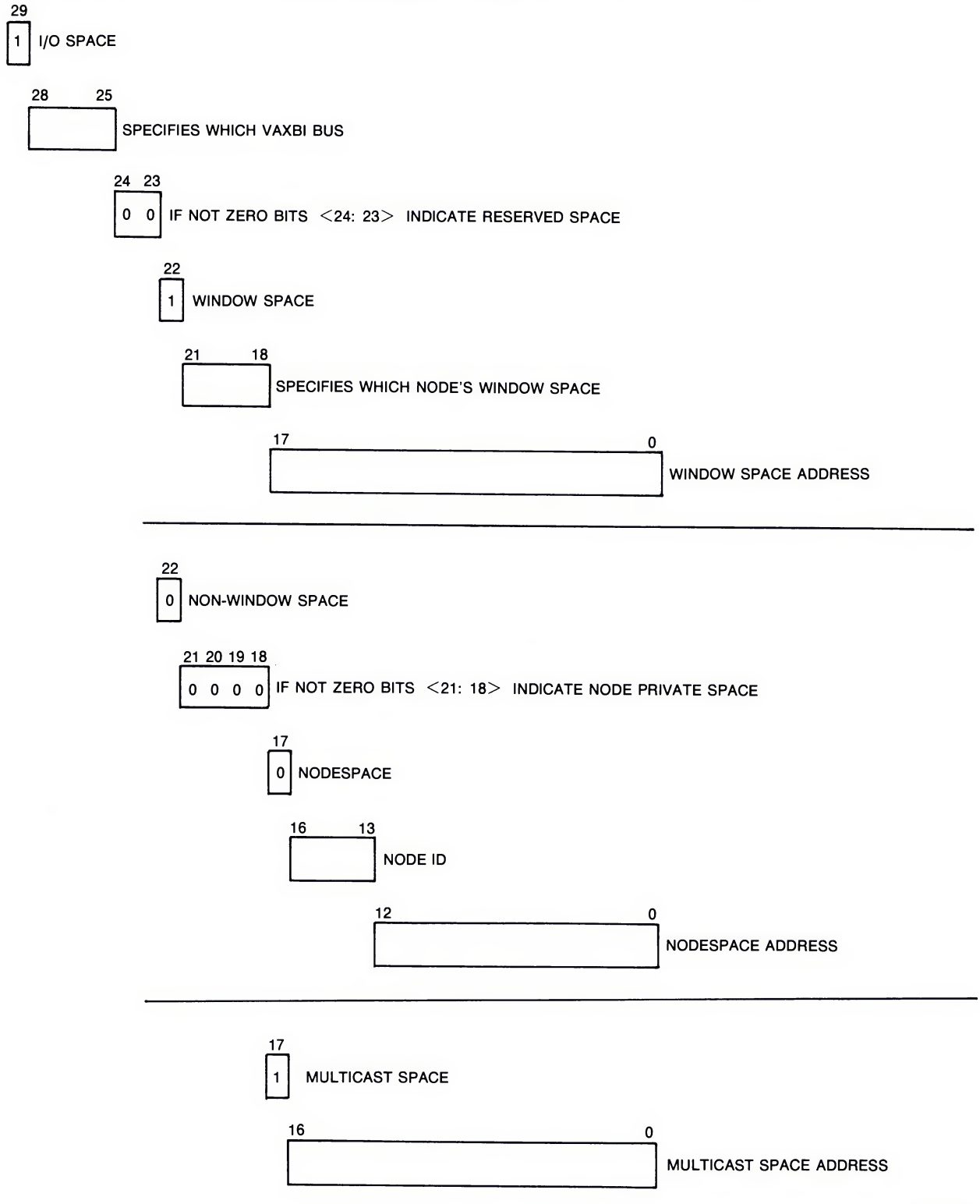


Figure B–5 Physical Addresses in VAXBI I/O Space



Nodespace	An 8KB block of addresses consisting of 256 bytes of *BIIC CSR space*, followed by *user interface CSR space*. A device can access the control and status registers (CSRs) of its backplane interconnect interface chip by using BIIC CSR space addresses. Device-specific registers reside in user interface CSR space. Because the VAX/VMS adapter initialization routine virtually maps nodespace for each VAXBI node on each VAXBI bus, a device driver can access both BIIC registers and device registers using virtual addresses. (See Sections B.4 and B.5 for a discussion of driver access to registers.)
Window space	A 256KB block used by a VAXBI adapter to map an I/O transfer to a target bus. Because VAX/VMS does not automatically map window space to virtual addresses, a driver that manipulates addresses in window space must itself allocate and fill sufficient system page-table entries for the range of its window space addresses. (See Section B.4.)

B.2.2 Backplane Interconnect Interface Chip (BIIC)

The *backplane interconnect interface chip* (BIIC) serves as the primary interface between the VAXBI bus and the user interface logic of a node. The BIIC supplies the logic necessary for a node to initiate and respond to transactions on the VAXBI bus, arbitrate bus ownership, send and receive interrupt requests, and monitor bus errors.

A node can enable, control, and monitor such activities by accessing the set of BIIC registers located in the first 256 bytes of its nodespace. Because the VAX/VMS adapter initialization routine virtually maps nodespace addresses, drivers for VAXBI devices can use virtual addresses to access BIIC registers. In addition, given the virtual address of the base of a device's nodespace, a driver can use the symbolic offsets, masks, and bit fields defined by the VMS macro $BIICDEF (in SYS$LIBRARY:LIB.MLB). Table B–1 describes these symbols.

B.3 Initialization Performed by VAX/VMS

During the phase of system initialization known as adapter initialization (INIADP) VAX/VMS performs a set of processor-specific tasks to identify and configure each device it discovers at each of the 16 nodes on each VAXBI bus in the system configuration.

The INIADP module configures DIGITAL-supplied and non-DIGITAL-supplied devices alike, performing the following activities as part of its initialization cycle:

1 Tests for the presence of a device at the node by issuing a MOVL instruction, the target of which is a system virtual address temporarily

mapped to the first longword of its nodespace. If this instruction is successful, it returns the contents of the BIIC Device Type Register of the addressed node to the processor.[3]

2 Records the 32-bit contents of the BIIC Device Type Register in the slot in the CONFREGL array that corresponds to the VAXBI bus and node at which it found the device,[4] and compares this value against a table of recognized device types.

3 If it *recognizes* the device, maps the number of pages specified in the table for the device type, and places the system virtual address of the base of the mapped nodespace in the slot in the SBICONF array that corresponds to the VAXBI bus and node at which it found the device.[5]

If it does *not* recognize the device, maps the entire 8KB of the node's nodespace into VAX/VMS virtual address space by allocating 16 system page table entries (SPTEs) and associating them with the 16 page-frame numbers (PFNs) of the physical addresses assigned to this node's nodespace on this VAXBI bus. INIADP then saves the base system virtual address of the resulting 8KB range in the longword slot corresponding to this node in the SBICONF array.

4 Performs such additional tasks as allocating and filling in data structures in a device-specific manner. For a non-DIGITAL-supplied device attached to a VAXBI bus, INIADP creates generic versions of the channel request block, interrupt dispatch block, and adapter control block—and fills in the appropriate vectors in the system control block—as discussed in Section B.3.1.

For devices it *recognizes*, INIADP additionally calls a VMS-supplied subroutine, the address of which it obtains from the device-type table, that performs further device-specific initialization.

For devices it does *not* recognize, INIADP must defer device-specific initialization to the device driver's initialization routine.

B.3.1 Data Structures

The INIADP module creates and prepares a channel request block, interrupt dispatch block, and an adapter control block in the manner described below. For each data structure it creates, INIADP fills in the first three longwords with the standard VAX/VMS header information (that is, the structure type, size, and links).

[3] If no device exists at a given VAXBI node address, the CPU becomes aware of this in a processor-dependent way. For example, the VAX 8200 and VAX 8300 processors experience a machine check, whereas the VAX 8500, VAX 8550, VAX 8700, and VAX 8800 processors determine that the node is vacant by reading an NXM (nonexistent memory) error from the BIIC Bus Error Register of the NBIB adapter on the VAXBI being examined.

[4] The CONFREGL array is a set of longwords in system pool pointed to by EXE$GL_CONFREGL. The CONFREGL array contains an entry for each possible VAXBI node. For VAX 8200 and VAX 8300 systems, with one VAXBI, this array has 16 entries. For VAX 8500, VAX 8550, VAX 8700, and VAX 8800 systems, this array has 16 entries for each VAXBI bus on the system.

[5] The SBICONF array is a set of longwords, similar in structure to the CONFREGL array and pointed to by MMG$GL_SBICONF, that lists the system virtual addresses of the base of the nodespace for each node on a VAXBI bus.

Channel Request Block

For the newly created channel request block (CRB), INIADP performs the following tasks:

- Sets up the resource wait queue header (CRB$L_WQFL and CRB$L_WQBL).
- Initializes the two interrupt dispatchers (CRB$L_INTD and CRB$L_INTD2) so that they have the effect of pushing general registers R0 through R5 onto the stack, and issuing a JSB instruction, the destination of which is, at initialization, a standard null interrupt handler which merely dismisses the interrupt. Later, when the specific device driver is loaded for the device (see Section B.7), the driver's interrupt-servicing routine address replaces this null interrupt handler in the dispatchers.

Interrupt Dispatch Block

INIADP initializes the interrupt dispatch block (IDB) in the following manner:

- Sets the number of device units controlled by this interrupt dispatch block (IDB$W_UNITS) to 1. The list of unit-control block (UCB) addresses in this IDB, as a result, is one longword in size. The driver-loading procedure writes a UCB address into this longword whenever it creates a new UCB associated with the controller. Because there is only one slot in this array, drivers for non-DIGITAL-supplied multidevice controllers must use a different mechanism to locate the UCB of interest at the time of an interrupt.
- Copies the virtual address of the base of this device's nodespace to IDB$L_CSR from the corresponding slot in the SBICONF array.

Adapter-Control Block

INIADP creates a truncated adapter control block (ADP) for a non-DIGITAL-supplied VAXBI device (48 bytes as opposed to the traditional 600 bytes). The ADP it creates contains no fields reserved for the allocation and accounting of data paths or mapping registers. INIADP prepares this generic ADP in the following manner:

- Copies the virtual address of the base of this device's nodespace to ADP$L_CSR from IDB$L_CSR.
- Places the VAXBI node ID of this device in ADP$W_TR.
- Stores the value AT$_GENBI (signifying the *generic VAXBI* ADP type) in ADP$W_ADPTYPE. Symbol AT$_GENBI has the value 13_{10} in VAX/VMS Version 4.5.
- Inserts the address of the new channel request block in ADP$L_CRB.
- Calculates the address of the first of the four interrupt vectors for this node in the system control block (SCB), and places it in ADP$L_AVECTOR. A driver can determine the addresses of the other three SCB vectors by adding 64, 128, or 192, respectively, to the address of this first SCB vector.
- Saves the offset of this first SCB vector from the start of its SCB page in ADP$W_BI_VECTOR. (Refer to Section B.3.2 for a description of the SCB.)

- Places in ADP$L_BI_IDR a longword mask with a single bit set, as appropriate to the VAX processor, that specifies which VAXBI node should become the destination of interrupts from this node. On VAX 8200 and VAX 8300 systems, the VAXBI node of the primary processor becomes the destination for interrupts; on VAX 8500, VAX 8550, VAX 8700, and VAX 8800 systems, it is the VAXBI node at which the NBIB adapter for this VAXBI bus resides.
- Stores in ADP$L_MBASCB—and in each of the device's four SCB vectors—the address of the interrupt dispatcher. The actual stored value is CRB$L_INTD+1, the set low bit of the address indicating that the interrupt stack be used to service the interrupt. Certain powerfail recovery operations use the contents of ADP$L_MBASCB to refresh the SCB vectors.
- Saves in ADP$L_MBASPTE the contents of the first of the 16 SPTEs that map the device's nodespace. Certain recovery operations use the contents of ADP$L_MBASPTE to restore correct SPTE values and remap nodespace following a power failure.

B.3.2 System Control Block

The system control block (SCB) consists of two or more pages of vectors. For all VAX processors, the first half page contains vectors used in exception dispatching. VAX/VMS uses the remainder of the first page, as well as subsequent pages, in a processor-dependent way.

For VAX 8200 and VAX 8300 systems, VAX/VMS assigns the vectors from 100_{16} to $1FC_{16}$ to VAXBI devices in the order of their node IDs.

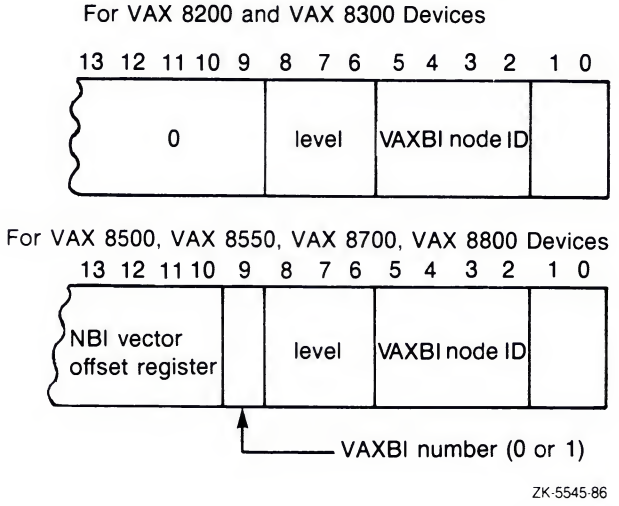
In contrast, VAX 8500/8550/8700/8800 system architecture relegates vectors 100_{16} to $1FC_{16}$ to NMI nexus vectors. Page 1 is reserved for the first "offsettable" device that exists in the system. (An "offsettable" device is an adapter, such as the BI-to-UNIBUS adapter (BUA), that passes interrupts from devices on another bus to the VAXBI and, from there, to the NMI and the processor.) If there is more than one "offsettable" device, an additional SCB page is needed for each.

Ultimately, the vectors for other devices attached to each of the four possible VAXBI buses of the system are contained in the four corresponding SCB pages from page 28 to page 31. Vectors for devices connected to VAXBI 0 and VAXBI 1 on NBI 0 are assigned to pages 28 and 29 of the SCB, respectively; vectors for devices connected to VAXBI 0 and VAXBI 1 on NBI 1 are likewise assigned to pages 30 and 31.

Generally, a VAX processor obtains a device vector from the BIIC registers of the node that has requested the interrupt (see Figure B–6). Information supplied in the device vector allows the processor to index to the corresponding interrupt-dispatching vector in the appropriate page of the SCB. For VAX 8200 and VAX 8300 systems, such information includes the interrupt level of the device and its VAXBI node ID. A similar vector for VAX 8500, VAX 8550, VAX 8700, and VAX 8800 devices further specifies the appropriate NBI vector offset and the number of the VAXBI bus.

The specific SCB interrupt-dispatching vector, thus found, transfers control to the interrupt-dispatching code in the device's CRB. Upon an interrupt from this device, the SCB vector will direct flow into the interrupt dispatcher, in the CRB, which will save the register contents and dispatch to the interrupt handler established by the device driver.

Figure B–6 VAXBI Device Vectors



B.4 Initialization Performed by the VAXBI Device Driver

All generic VAXBI device drivers must specify *GENBI* as the adapter type in the **adapter** argument to the DPTAB macro.

The device driver's initialization routines are expected to initialize the device-specific aspects of the VAXBI device. For non-DIGITAL-supplied devices, the initialization routines perform the sort of tasks that the INIADP module performs for the DIGITAL-supplied devices it discovers on a VAXBI bus. For single-unit devices, a separate unit-initialization routine may not be necessary.

The VAX/VMS System Generation Utility (SYSGEN) calls the controller-initialization routine at IPL 31, passing it the following values in the listed general registers:

- R4 pointing to the system virtual address of the device's nodespace
- R5 pointing to the IDB
- R6 pointing to the device data block
- R8 pointing to the CRB

After the controller-initialization routine has completed, SYSGEN calls the driver's unit-initialization routine at IPL 31, and passes it the following values in the listed general registers:

- R3 pointing to the system virtual address of the device's nodespace
- R5 pointing to the UCB

Hardware initialization might include such activities as writing values to BIIC and device-specific registers, examining the results of the BIIC self test, mapping a node's window space, building data structures to control the device, and linking these structures into chains of similar data structures.

This section provides some ideas and guidelines for code that may be necessary in an initialization routine. There is no requirement that driver code perform all of the functions discussed here. The needs of various devices differ, and some devices make more demands on driver software than others.

Code examples in the section assume that R4 initially contains the virtual address of the base of the device's nodespace and R8 contains the virtual address of the device's CRB.

B.4.1 Examining BIIC Self-Test Status

According to the hardware specification for all devices attached to a VAXBI bus, a VAXBI node undergoes a self test on power failure recovery and at system boot time. The BIIC indicates the successful completion of the self test by setting BIIC$V_STS and by clearing BIIC$V_BROKE in BIIC$L_BICSR.

A driver unit initialization routine should test these bits before performing any transaction on the VAXBI bus. If BIIC$V_STS is clear, then self test is still under way. If BIIC$V_BROKE is set, then the driver action is implementation-specific. In any event, a driver should not set UCB$V_ONLINE in UCB$L_STS if the node is not usable.

The maximum duration of the BIIC self test is ten seconds. If a VAXBI node implements the maximum self-test time, then the driver unit initialization routine may have to spinwait for the setting of BIIC$V_STS (for instance, by embedding the testing instructions in an invocation of the TIMEDWAIT macro). Driver unit initialization routines should perform this spinwait only when UCB$W_POWER in UCB$L_STS is set. Otherwise, the driver is being loaded by SYSGEN, and a long spinwait at high IPL will have adverse effects on the rest of the VMS system.

Normally, only diagnostics initiate a self test by setting the SST bit in the BIIC. A VAXBI driver that sets this bit must take special precautions to avoid a machine check and to avoid undetected corruption of VAXBI memory. These precautions include the following steps:

1. Begin a machine check protection block (using the $PRTCTINI macro). Code within the block executes at IPL 31.
2. Disable arbitration on the VAXBI node being reset.
3. Set BIIC$V_SST and BIIC$V_STS simultaneously to initiate the self test.
4. Do not set SST in the same instruction that disables arbitration.
5. End the machine check protection block (using the $PRTCTEND macro).
6. Do not access the BIIC registers for at least one microsecond. You may not even check the state of the STS bit during this interval.
7. Do not access any other address on the VAXBI node until the self test has completed.

B.4.2 Clearing BIIC Errors, Setting Interrupts, and Enabling Interrupts

There is a set of tasks that a VAXBI driver should perform during initialization that ensures that interrupts are properly enabled and delivered to an appropriate VAXBI target node. These tasks include the following:

- Clearing any outstanding set bits in the Bus Error Register.
- Setting the target node for interrupts in the Interrupt Destination Register.
- Setting the device interrupt vector in the Error Interrupt Control Register.
- Setting the device interrupt vector in the User Interface Interrupt Control Register.
- Enabling hard and soft error interrupts as required by the device. Typically hard errors and enabled and soft errors are disabled.
- Enabling interrupts upon certain types of transactions to user interface CSR space.

It is important that the interrupt vectors and destination be set up *before* BIIC hard error and soft error interrupts are enabled. An error occurring while error interrupts are enabled but the vector uninitialized could lead to an invalid condition.

B.4.2.1 Clearing the Bus Error Register

The following example clears all set bits in the Bus Error Register (BIIC$L_BER) to prevent spurious or pending error interrupts at initialization.

```
MOVL   BIIC$L_BER(R4), BIIC$L_BER(R4)      ;Clear all set write-1-to-clear
                                           ; bits in BIIC$L_BER
```

B.4.2.2 Loading the Interrupt Destination Register

The Interrupt Destination Register (BIIC$L_IDR) specifies which VAXBI node should become the destination of interrupts from this node. On the VAX 8200 and VAX 8300 systems, the VAXBI node of the primary CPU becomes the destination for interrupts. On VAX 8500, VAX 8550, VAX 8700, and VAX 8800 systems, the VAXBI node of the NBIB on the particular VAXBI on which this device resides becomes the destination for such interrupts.

The VAX/VMS system initialization procedure described in Section B.3 creates a 32-bit mask with the appropriate bit set and stores it in ADP$L_BI_IDR. If a driver must set the Interrupt Destination Register, it can simply move this value to the BIIC register:

```
MOVL   CRB$L_INTD+VEC$L_ADP(R8),R0        ;Get ADP address
MOVL   ADP$L_BI_IDR(R0),BIIC$L_IDR(R4)    ;Write to IDR
```

B.4.2.3 Setting Interrupt Vectors

A VAXBI node uses the Error Interrupt Control Register (BIIC$L_EICR) to determine the SCB vector through which to interrupt when a BIIC at this node detects a bus error. The User Interface Interrupt Control Register (BIIC$L_UICR) similarly controls the operation of interrupts initiated by the device at this node.

Because the VAX/VMS system initialization procedure described in Section B.3 saves the offset of the node's first SCB vector from the start of its SCB page in ADP$W_BI_VECTOR, a driver can initialize both of these registers by using code similar to that in the following example:

```
MOVL    CRB$L_INTD+VEC$L_ADP(R8),R0        ;Get ADP address
MOVZWL  ADP$W_BI_VECTOR(R0),R2             ;Get device vector
MOVL    BIIC$L_UICR(R4),BIIC$L_UICR(R4)    ;Clear user vector
MOVL    R2,BIIC$L_UICR(R4)                 ;Set user vector
BISL    #1@<BIIC$V_LEVEL+BIIC$S_LEVEL-1>,R2
                                           ;OR in interrupt level
                                           ;BR7 in this case
MOVL    BIIC$L_EICR(R4),BIIC$L_EICR(R4)    ;Clear error vector
MOVL    R2,BIIC$L_EICR(R4)                 ;Set error vector
```

Note that the driver clears both vectors before it actually sets them. Clearing BIIC$L_UICR and BIIC$L_EICR causes any pending interrupt to be cleared. Also note that the interrupt level must be set in BIIC$L_EICR; in this case BR7. If the level is not set, an error interrupt will never be generated.

B.4.2.4 Enabling Error Interrupts

Finally, to enable interrupts that report errors detected by the node's BIIC, the controller-initialization routine can set the soft error interrupt-enable or hard error interrupt-enable bits in the VAXBI Control and Status Register. The BIIC sets bits in the Bus Error Register (BIIC$L_BER) to reflect the type of bus error reported by the interrupt.

```
BISL    #<BIIC$M_SEIE!BIIC$M_HEIE>,-       ;Soft error interrupt enable
        BIIC$L_BICSR(R4)                   ;Hard error interrupt enable
```

B.4.2.5 Enabling BIIC Options

Device registers are in the area of nodespace called user interface CSR space, and are located following the 256 bytes reserved for the BIIC-required registers. Use of user interface CSR space is implementation-dependent.

For the processor to be alerted to various transactions directed at user interface CSR space, the controller-initialization routine of devices that support such transactions should set appropriate bits in the BCI Control and Status Register (BIIC$L_BCICR). See Table B–1 for definitions of these bits.

The following example enables a node to alert the node specified as the interrupt destination (in BIIC$L_IDR) when a retry timeout, STOP command, or read or write transaction is directed at its user interface CSR space.

```
BISL    #<BIIC$M_STOPEN!-                  ;Stop enable
          BIIC$M_RTOEVEN!-                 ;Retry timeout enable
          BIIC$M_UCSREN>,-                 ;User CSR enable
          BIIC$L_BCICR(R4)
```

B.4.3 Mapping Window Space

Each VAXBI, starting at address 20400000_{16}, provides 16 address blocks of 256K bytes apiece, called *window space*. VAXBI nodes can use window space if it is necessary to map VAXBI transactions to memory space on a target bus, although few nodes use this feature.

Whereas the VAX/VMS initialization routine maps each VAXBI node's nodespace to virtual addresses, it does not automatically map each node's window space. If a device needs to use its window space, it is up to the driver's unit-initialization routine to map this space.

First of all, the driver must determine the starting physical address of the node's window space. Figure B–5 illustrates how VAXBI addresses are constructed. Drivers can use the following VAX/VMS-supplied macros (in SYS$LIBRARY:LIB.MLB) to access pertinent VAXBI addresses and values:

$IO8SSDEF (for the VAX 8200 and VAX 8300 systems)
$IO8NNDEF (for the VAX 8500, VAX 8550, VAX 8700, and VAX 8800 systems)

To determine the starting address of a node's window space, the driver should perform the following actions:

1. Extract the VAXBI node ID from BIIC$L_BICSR.
2. Multiply the node ID with the size of window space, as stored in IO8SS$AL_NDSPER for VAX 8200 and VAX 8300 systems and IO8NN$AL_NDSPER for VAX 8500, VAX 8550, VAX 8700, and VAX 8800 systems. VAXBI device drivers running on a VAX 8200 or VAX 8300 system can skip to step 4.
3. Perform steps necessary to account for the existence of multiple VAXBI buses on the system. These steps are only necessary for VAXBI device drivers running on a VAX 8500, VAX 8550, VAX 8700, or VAX 8800 system. They include the following:
   a. Determine which VAXBI bus the node is attached to by extracting the VAXBI bus number from bits <7:4> of ADP$W_TR.
   b. Multiply the VAXBI bus number thus obtained by 2000000_{16}, the amount of physical address space allocated for each VAXBI bus.
   c. Add the result to the product obtained in step 2.
4. Add to the accumulated calculations the base address of the start of window space for node 0 on a VAXBI bus. This address can be determined by adding the values contained in IO8SS$AL_NODESP and IO8SS$AL_IOBASE (for a VAX 8200 or VAX 8300 system) or IO8NN$AL_NODESP and IO8NN$AL_IOBASE (for a VAX 8500, VAX 8550, VAX 8700, or VAX 8800 system).

Secondly, each page of window space to be used must be associated with a system page-table entry (SPTE) that maps the page-frame number (PFN) of the physical page in window space to a system virtual address. VAX/VMS includes the routine IOC$ALLOSPT in module IOSUBNPAG that, given the number of SPTEs to be allocated in R1, returns in R2 the starting system virtual page number (SVPN) of the first allocated SPTE of the requested amount. (See Section B.8.2 for additional information on IOC$ALLOSPT.)

Because IOC$ALLOSPT expects to be called at IPL$_SYNCH, the unit-initialization routine must fork from IPL$_POWER to IPL$L_SYNCH before calling it. See Section B.4.4 for a discussion of forking in a driver initialization routine.

Finally, once the SPTEs have been allocated, the driver moves the physical page numbers (PFNs) of the window space pages into the SPTEs.

B.4.4 Forking from a Driver Initialization Routine

If a driver initialization routine must fork to perform a thread of code that must synchronize with code or a structure synchronized at a lower IPL, it must take special care to avoid breaking that synchronization.

First of all, because the System Generation Utility, under normal circumstances, immediately calls a driver's unit-initialization routine at IPL$_POWER after its controller-initialization completes, the unit-initialization routine must be prepared for the instance of a controller-initialization routine that forks. Such a unit-initialization routine would complete before the fork thread of the controller-initialization routine resumed.

A fork thread in a unit-initialization routine (or a controller-initialization routine in a driver without a unit-initialization routine) must otherwise take the following precautions to avoid breaking synchronization:

- Allocate a separate fork block within the UCB. Do not attempt to allocate this block with EXE$ALONPAGEDYN. The separate fork block prevents a conflict with the use of the normal UCB fork block by the IOFORK routine.
- Use a semaphore bit to protect against multiple forking. Remember that the unit initialization routine may be called repeatedly in the case of power failures. If the semaphore shows that a fork is in progress, then exit without attempting to fork.
- Invoke EXE$FORK with R5 pointing to the new fork block. Restore the original value of R5 once the fork process is active.
- Remember to restore all registers on exit to the unit initialization routine. Since EXE$FORK removes the caller's address from the stack and returns to the caller's caller, the unit initialization routine must set up a dummy caller's caller routine to restore registers destroyed by EXE$FORK.

B.5 DMA Transfers

The method by which a device accomplishes direct-memory-access (DMA) transfers depends upon the characteristics of the device. As part of a VAXBI read or write transaction, such a device must place on the VAXBI bus a physical address, the target of which is a memory node or a node (such as an NBIB adapter) that transmits the request to memory across another bus.

For the DMA device to successfully access the memory pages of a buffer involved in an I/O transfer, it must be given sufficient information as to the size and location of these buffer pages, the type of transaction that is requested, an offset into the first page of the buffer, and the length of the transaction. In addition, if the size of the transaction causes it to exceed the boundaries of a page, the device must have some means of accessing the remaining pages—even if they are, as is most likely, scattered throughout physical memory.

As a result, devices make use of several types of structures, the purpose of which is to help generate a succession of contiguous physical addresses on the VAXBI bus, that map to the various pages of the buffer involved in the transfer. Some possible constructions of this sort include the following:

- A physically contiguous buffer in memory
- System page tables in system memory

- Process page tables locked in system memory
- Mapping registers in the device's VAXBI I/O address space

A separate but related issue results from the fact that the original buffer, as specified in the user Queue-I/O request, is in process space and is mapped by process page-table entries. Because the driver cannot rely on process context existing at the time the device is ready to service the I/O request, it must have some means of guaranteeing that it can access both the data involved in the transfer and the page-table entries that map the buffer.

VAX/VMS supplies two separate techniques, applied by traditional VAX/VMS drivers and described in full in the *Writing a Device Driver for VAX/VMS* manual.

- *Direct I/O*, the technique used most commonly by DMA drivers, locks the user buffer in memory as well as the page-table entries that map it. The function-decision table (FDT) of such a driver calls a VAX/VMS-supplied FDT routine that prepares the user buffer for direct I/O.
- *Buffered I/O* is the strategy whereby the driver FDT dispatches to an FDT routine in the driver that allocates a buffer from nonpaged pool. It is this intermediate buffer that is involved in the DMA transfer. Driver preprocessing routines copy the data from the user buffer to the system buffer for a write request; I/O postprocessing routines deliver data from the system buffer to the user buffer for a read request.

That DMA drivers may make use of either VMS direct-I/O or buffered-I/O is one way by which these drivers can supply specific information needed by the device to accomplish a DMA transfer. Those driver FDT routines that call a VAX/VMS direct-I/O FDT routine leave the following information in the device's unit-control block (UCB):

UCB$L_SVAPTE	Virtual address of the system page-table entry (PTE) for the first page used in the transfer
UCB$W_BOFF	Byte offset in the first page of the transfer buffer
UCB$W_BCNT	Size in bytes of the transfer

FDT routines that elect buffered-I/O call EXE$ALLOCBUF to obtain a nonpaged pool buffer and initialize the same UCB fields with the following information:

UCB$L_SVAPTE	Virtual address of system buffer used in the I/O transfer
UCB$W_BOFF	Number of bytes to be charged to the process for the transfer
UCB$W_BCNT	Size in bytes of the transfer

If a driver's fork process must manipulate the data in any way at fork level (that is, outside of the driver's FDT routines), then it needs a virtual address it can use to access the data. Typically this is done by using a nonpaged pool buffer. It can also be done by loading a system page-table entry with the correct PFN and computing the associated system virtual address. The drivers for the disks that have ECC correction do this when there is an ECC error detected. The controller can tell the driver that the error in the data in memory can be corrected by applying some pattern to a part of the data, but the fork process has to perform the correction, not the controller.

```
MOVL    IRP$L_SVAPTE(R3),R2             ;Get address of system buffer
SUBW3   #12,8(R2),UCB$W_BCNT(R5)        ;Calculate system buffer length
BICW3   #C^<VA$M_BYTE>,(R2),UCB$W_BOFF  ;Put offset in buffer
EXTZV   #VA$V_VPN,#VA$S_VPN(R2),R2      ;Get system virtual page number
MOVL    G^MMG$GL_SPTBASE,R1             ;Get address of system page table
MOVAL   (R1)[R2],UCB$L_SVAPTE(R5)       ;Get system virtual address of page
```

B.5.1 Example: DMB32 Asynchronous/Synchronous Multiplexer

The DMB32 asynchronous/synchronous multiplexer can use any of four different modes of address translation for DMA accesses. Under each of these modes, the DMB32 requires that its driver supply an address by which it can either directly or indirectly obtain the pages of the buffer that is involved in the transfer. The four different translation modes require such addresses in one of the following forms:

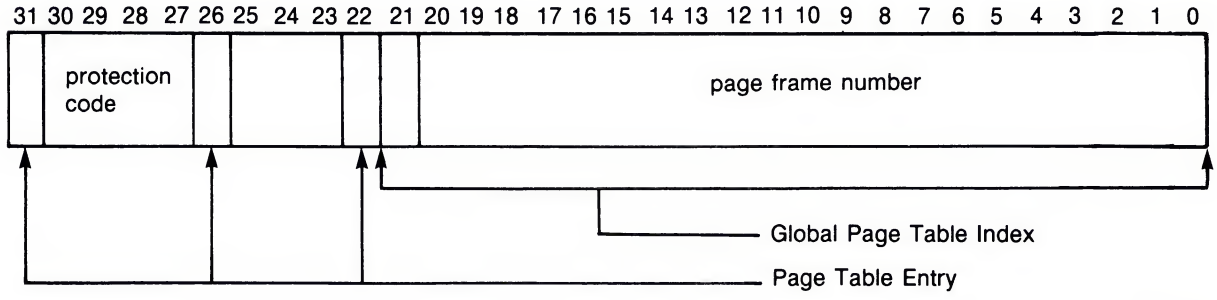
1. System virtual address of a buffer
2. System virtual address of a page-table entry
3. Physical address of a page table
4. Address of a physically contiguous buffer

System Virtual Address of a Buffer and System Virtual Address of a Page-Table Entry

The DMB32 can itself perform the first two types of address translation because it can read entries in the VAX/VMS system page table (see Figure B–7, as well as the *VAX/VMS Internals and Data Structures* manual). The controller-initialization routine of a DMB32 device driver supplies the physical address and length of the VAX/VMS system page table, plus the virtual address and length of the VAX/VMS global page table. It also sets a page-table-valid bit in a device maintenance register.

As a result, a driver for a DMB32 device could use either direct-I/O or buffered-I/O, and accordingly load a device register with the system virtual address of the page-table entry that maps the buffer or the system virtual address of the buffer itself. After the driver has loaded other device registers with a buffer offset value and a transfer size—and set the "start" bit in a DMB32 line-control register—the DMB32 performs the transfer without any additional mapping or other driver intervention.

Figure B–7 Page Table Entry



ZK-5544-86

Physical Address of a Page Table

In this mode, the DMB32 can be given the physical address of a page table that maps the I/O transfer. The DMB32 architecture mandates that each page-table entry be four bytes long and that the page table be aligned on a longword boundary. Also, each page is 512 bytes long. However, the page table can be anywhere in memory, possibly at a range of VAXBI I/O-space addresses belonging to the node to which the DMB32 adapter is attached. To perform a DMA transfer under this addressing mode, the DMB32 adapter requires the offset of the first byte of the buffer which is in the page described by the page-table entry. Each page-table entry contains bits <29:9> of the physical address of the page that is to be accessed.

In this case, the driver must extract the PFNs of the pages involved in the transfer and insert them into the page table of the device. The following is an example of a routine that translates a system virtual address to a physical address. It returns the physical address at the top of the stack.

```
VIRT_TO_PHYAD:
     PUSHL   (SP)                          ;Create slot at top of stack for return
                                           ; value
     PUSHR   #^M<R0,R1,R2,R3>              ;Save registers
     BICL3   #-512,R1,R0                   ;R0 = byte offset of address
     EXTZV   #VA$V_VPN,-                   ;Extract VPN
             #VA$S_VPN,R1,R2               ; and put it in R2
     MOVL    G^MMG$GL_SPTBASE,R3           ;R3 => system page table
     MOVL    (R3)[R2],R3                   ;R3 => PTE
     EXTZV   #PTE$V_PFN,-                  ;Get page frame number of buffer
             #PTE$S_PFN,R3,R3              ; page into R3
     ASHL    #VA$V_VPN,R3,R3               ;Shift into place for physical address
     BISL3   R0,R3,20(SP)                  ;Put result into stack slot
     POPR    #^M<R0,R1,R2,R3>              ;Restore registers
     RSB                                   ;Return to caller
```

Physical Address of a Buffer

If the device can neither read system page tables nor has its own scatter-gather map—and must perform a DMA transfer that spans physical pages—it must rely upon the actual contiguity of the physical pages involved in the transfer. Because there is no guarantee that this is the state of the user's buffer, the driver must allocate an intermediate buffer consisting of contiguous physical pages. The driver never deallocates this buffer unless the driver is being unloaded (by means of SYSGEN's RELOAD command). The best time to allocate such a buffer is during the device's initialization, when memory is most likely to be contiguous.

The VAX/VMS routine EXE$ALOPHYCNTG, described in the *Writing a Device Driver for VAX/VMS* manual, allocates such a buffer. The size of the buffer that should be allocated depends on the device's characteristics and the size of the transfers requested on the device. A buffer of four pages is likely to be large enough for most disk transfers, for example; but if you have enough memory on your system, you might want to make your buffer the size of a disk track in order to reduce disk latency. In any event, large transfers to the device can be segmented into transfers the size of your intermediate buffer.

The start-I/O routine of such a driver copies the data from the user's buffer into the intermediate, physically contiguous buffer by means of the routine IOC$MOVFRUSER.

The driver then sets up the device for the DMA transfer:

1. Determines the physical address of the buffer from the system virtual address returned by EXE$ALOPHYCNTG
2. Moves the address to the device address register
3. Activates the device
4. If the transfer size exceeds the size of the buffer, returns to step 1

When a user requests a transfer from such a device, the driver moves the data from the device to the intermediate, physically contiguous buffer by means of a DMA transfer, then calls IOC$MOVTOUSER to copy the data into the user's buffer.

B.6 Register-Dumping Routine

In the event of a device error or a VAXBI bus error, a driver's register-dumping routine should contain code that makes certain interesting registers available for error logging. Apart from any device registers that should be saved, the following BIIC registers may contain information important in determining the cause of the error: the Device Register (BIIC$L_DTREG), the VAXBI Control and Status Register (BIIC$L_BICSR), the Bus Error Register (BIIC$L_BER), the Error Interrupt Control Register (BIIC$L_EICR), and the Interrupt Destination Register (BIIC$L_IDR).

The following is an example of part of a register-dumping routine that pushes the contents of these BIIC registers into an error buffer.

```
MOVL    BIIC$L_DTREG(R4),(R0)+          ;Device Type Register
MOVL    BIIC$L_BICSR(R4),(R0)+          ;BIIC CSR Register
MOVL    BIIC$L_BER(R4),(R0)+            ;Bus Error Register
MOVL    BIIC$L_EICR(R4),(R0)+           ;Error Interrupt Control Register
MOVL    BIIC$L_IDR(R4),(R0)+            ;Interrupt Destination Register
```

B.7 Loading a VAXBI Device Driver

The System Generation Utility (SYSGEN) loads the device driver into system virtual memory, creates additional data structures for the device unit, connects the device's interrupt vectors, and calls the device driver's controller-initialization routine and unit-initialization routine.

The *Writing a Device Driver for VAX/VMS* manual discusses the SYSGEN commands commonly used during driver loading. The following discussion pertains to those aspects of the loading process that specifically relate to the support of non-DIGITAL-supplied VAXBI devices.

Traditionally, SYSGEN is invoked near the end of system initialization processing during the execution of the system startup command procedure (SYS$SYSTEM:STARTUP.COM). STARTUP.COM generally issues a SYSGEN AUTOCONFIGURE command, the result of which is that SYSGEN scans various device tables to determine devices VAX/VMS expects to be connected to each VAXBI bus configured in the system. Ultimately, as the autoconfigure facility discovers the data structures associated with VAXBI devices recognized by VAX/VMS, it loads the associated device drivers and invokes their initialization routines.

Because the autoconfigure facility cannot recognize non-DIGITAL-supplied VAXBI devices, STARTUP.COM (or a later invocation of SYSGEN) must explicitly request that SYSGEN connect the device.[6] SYSGEN responds to such explicit requests by utilizing the data structures created by the INIADP module for the unknown VAXBI device to load the associated device driver and invoke its initialization routines.

For example, suppose that an unknown VAXBI device were located at node 3 on a given VAXBI bus, and that the software device driver for this device were known as "ZZDRIVER". During INIADP processing, VAX/VMS would have encountered an unknown type of VAXBI device at node 3 and would have performed the following operations:

- Mapped the nodespace for node 3 into system virtual memory
- Constructed various data structures to govern the future operation of this device

SYSGEN executes in response to the following commands:

```
$ RUN SYS$SYSTEM:SYSGEN
SYSGEN> CONNECT ZZA0:/ADAPTER=3
```

SYSGEN performs the following activities:

1. Searches the list of ADPs in the system to find the ADP for this VAXBI node (node 3) and, in turn, locates the corresponding CRB and IDB by following pointers in the ADP.
2. Loads ZZDRIVER into system virtual memory. If the /DRIVER qualifier is specified, SYSGEN loads the specified driver instead.
3. Creates a UCB for device *ZZA0:* and places the address of the device's CRB in that UCB. SYSGEN also initializes other UCB fields at this time.
4. Sets the first entry in the IDB UCB array (IDB$L_UCBLST) to point to the new UCB.
5. Creates a DDB for the *ZZA* device/controller combination. This allows user programs to assign I/O channels to device *ZZA0:* later. This DDB, in turn, points to the location in memory where *ZZDRIVER* has been loaded and to the UCB for the *ZZA0:* device.
6. Calls the controller-initialization routine in *ZZDRIVER* at IPL 31.
7. Calls the unit-initialization routine in *ZZDRIVER* at IPL 31.

Note: If you did not specify *GENBI* as the adapter type in the adapter argument to the DPTAB macro, the CONNECT command will fail with the error message:

```
%SYSGEN-E-INVVEC, invalid or unspecified interrupt vector
```

B.8 Reference Material

The following sections include reference material concerning the contents of the BIIC register set and the routines discussed in this document.

[6] Because the autoconfigure facility will never be called for a non-DIGITAL-supplied device, any unit-delivery routine that a VAXBI device driver may include will never be called.

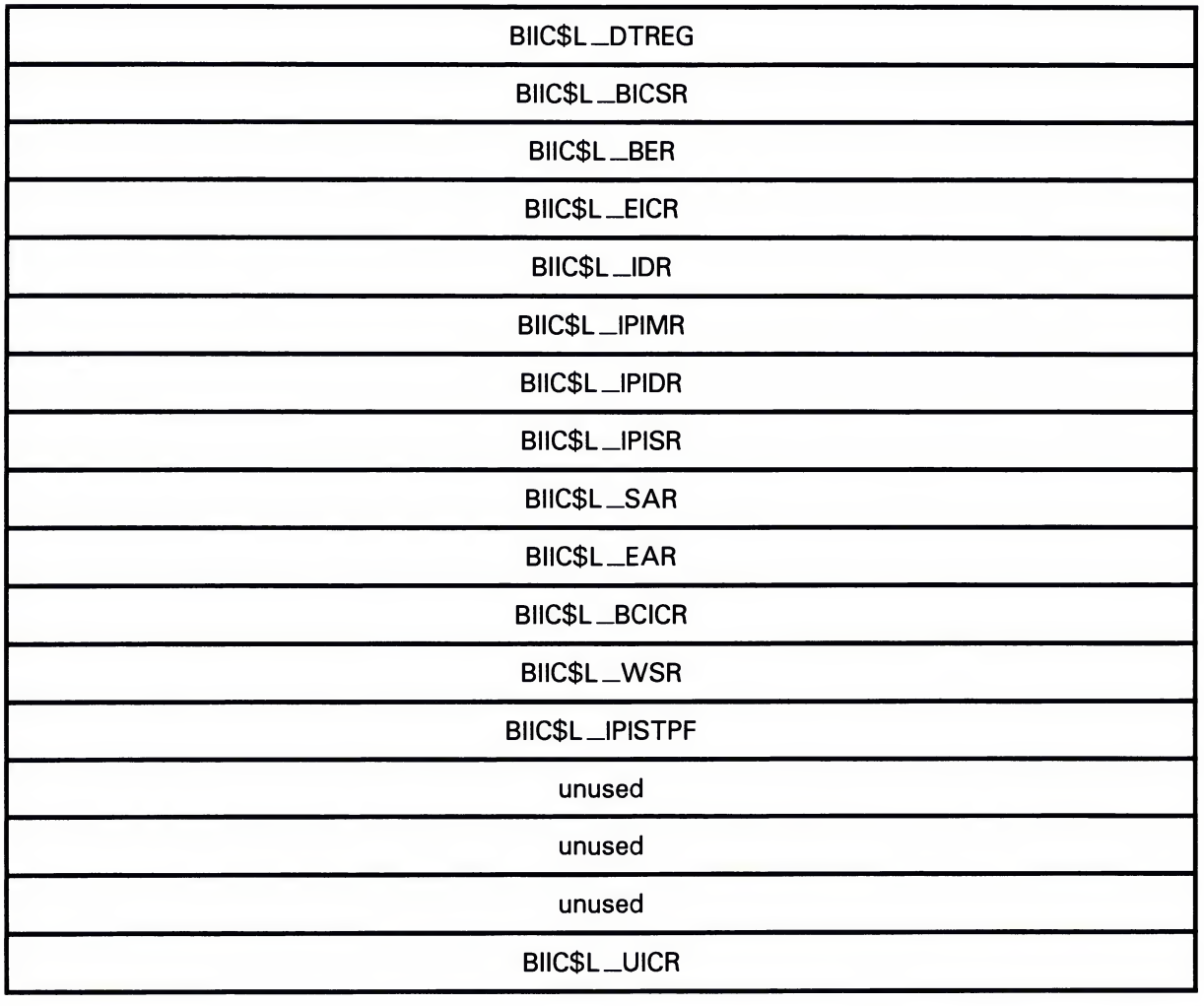
B.8.1 BIIC Register Definitions

Each VAXBI node is required to implement a minimum set of registers contained in specific locations within the node's nodespace. VAX/VMS automatically maps each node's nodespace at boot time and provides the macro $BIICDEF (in SYS$LIBRARY:LIB.MLB) to define offsets to the BIIC registers and their significant bit fields.

The contents of the BIIC registers are illustrated in Figure B–8 and described in Table B–1. See the *VAXBI Options Handbook* for a discussion of the BIIC and the rules for configuring its registers.

Note: Fields marked "Reserved to DIGITAL" are reserved for DIGITAL's future use and should contain zeros.

Figure B–8 Backplane Interconnect Interface Chip (BIIC) Registers



(Continued on next page)

Figure B–8 (Cont.) Backplane Interconnect Interface Chip (BIIC) Registers

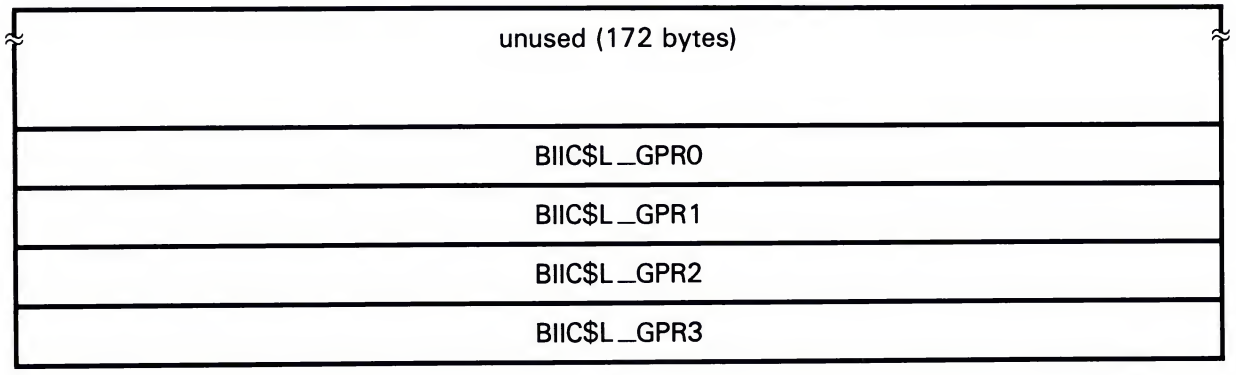


Table B–1 Contents of the BIIC Registers

Field Name	Contents	
BIIC$L_DTREG	Device Register. BIIC$L_DTREG consists of the following two words:	
	BIIC$W_DEVTYPE	Device type. This field is written by device hardware and self-test microcode. It contains two bit fields: BIIC$V_MEMNODE (bits <14:8>) when clear, indicates a memory node. BIIC$V_NONDEC (bit 15), when clear indicates a DIGITAL-supplied device; it should be 1 otherwise.
	BIIC$W_REVCODE	Revision code.
BIIC$L_BICSR	VAXBI Control and Status Register. The following fields are defined within BIIC$L_BICSR.	

Bit Field	Contents
BIIC$V_NODE_ID[1]	Node ID. This field is automatically loaded during the power-up sequence. Reserved to DIGITAL.
BIIC$V_ARBCNTL	Arbitration mode used by the node. Currently, all arbitration modes except dual round-robin arbitration are reserved to DIGITAL. Correspondingly, these two bits should be clear. When these two bits are set, arbitration is disabled, thus preventing a node from starting a VAXBI transaction.

[1]Read-only field.

Table B–1 (Cont.) Contents of the BIIC Registers

Field Name	Contents

Bit Field	Contents
BIIC$V_SEIE	Soft error interrupt enable. When set, this bit allows the node to generate an interrupt when the soft error summary bit (BIIC$V_SES) in this register is set.
BIIC$V_HEIE	Hard error interrupt enable. When set, this bit allows the node to generate an interrupt when the hard error summary bit (BIIC$V_HES) in this register is set.
BIIC$V_UWP	Unlock write pending. When set, this bit signals that the master port interface at this node has successfully completed an IRCI (Interlock Read with Cache Intent) transaction. The node clears this bit when it successfully completes a corresponding UWMCI (Unlock Write Mask with Cache Intent) instruction.
<9> [1]	Reserved to DIGITAL. Must be zero.
BIIC$V_SST	Node reset. This bit is normally used by diagnostics to initiate the BIIC internal self test. Prior to initiating a BIIC self test, a node should disable arbitration by setting both bits in BIIC$V_ARBCNTL. When BIIC$V_SST is set, the self-test status bit (BIIC$V_STS) in this register must also be set. Reads to BIIC$V_SST return a zero.
BIIC$V_STS	Self-test status. When set, this bit indicates that the BIIC has passed its self test. The controller-initialization routine of a VAXBI device driver should inspect this bit and the BIIC$V_BROKE bit before proceeding with any VAXBI transactions. During the self-test sequence, BIIC$V_STS will automatically be reset by the BIIC to allow the proper recording of the new self-test results at the end of self test.
BIIC$V_BROKE[2]	Broke bit. When cleared by the device's self test, this bit indicates that device has passed its self test. The controller-initialization routine of a VAXBI device driver should inspect this bit and the BIIC$V_STS bit before proceeding with any VAXBI transactions.
BIIC$V_INIT[2]	Initialization bit.
BIIC$V_SES[1]	Soft error summary. When set, this bit indicates that one or more of the soft error bits in the Bus Error Register (BIIC$L_BER) is set.
BIIC$V_HES[1]	Hard error summary. When set, indicates that one or more of the hard error bits in the Bus Error Register (BIIC$L_BER) is set.

[1] Read-only field.

[2] Write-one-to-clear bit. Write-type transactions cannot set this bit.

Table B–1 (Cont.) Contents of the BIIC Registers

Field Name	Contents

Bit Field	Contents
BIIC$V_BIICTYPE[1]	BIIC type. These bits <23:16> always contain 00000001.
BIIC$V_BIICREVN[1]	BIIC revision number.

BIIC$L_BER

Bus Error Register.

The following bits are defined within BIIC$L_BER. Bits <30:16> are hard error bits and bits <2:0> are soft error bits.

Bit Field	Contents
BIIC$V_NPE[2]	Null bus parity error.
BIIC$V_CRD[2]	Corrected read data.
BIIC$V_IPE[2]	ID parity error.
BIIC$V_UPEN[1]	User parity enabled.
<14:4> [1]	Reserved to DIGITAL. Must be zero.
BIIC$V_ICE[2]	Illegal confirmation error.
BIIC$V_NEX[2]	Nonexistent address.
BIIC$V_BTO[2]	Bus timeout.
BIIC$V_STO[2]	Stall timeout.
BIIC$V_RTO[2]	Retry timeout.
BIIC$V_RDS[2]	Read data substitute.
BIIC$V_SPE[2]	Slave parity error.
BIIC$V_CPE[2]	Command parity error.
BIIC$V_IVE[2]	IDENT vector error.
BIIC$V_TDF[2]	Transmitter during fault.
BIIC$V_ISE[2]	Interlock sequence error.
BIIC$V_MPE[2]	Master parity error.
BIIC$V_CTE[2]	Control transmit error.
BIIC$V_MTCE[2]	Master transmit check error.
BIIC$V_NMR[2]	NO ACK to multiresponder command received.
<31> [1]	Reserved to DIGITAL. Must be zero.

[1]Read-only field.

[2]Write-one-to-clear bit. Write-type transactions cannot set this bit.

Table B–1 (Cont.) Contents of the BIIC Registers

Field Name	Contents
BIIC$L_EICR	Error Interrupt Control Register. This register supplies information the node uses to request and monitor the status of both BIIC-detected and forced-error interrupts: that is, those interrupts signaled by either the setting of a bit in the Bus Error Register (BIIC$L_BER) or the setting of the force bit (BIIC$V_EIFORCE) in this register, respectively. The node can initiate BIIC-detected error-interrupt requests only if the appropriate error-interrupt enables (BIIC$V_SEIE and/or BIIC$V_HEIE) are set in the VAXBI Control and Status Register (BIIC$L_BICSR). The following fields are defined within BIIC$L_EICR.

Bit Field	Contents
<1:0> [1]	Reserved to DIGITAL. Must be zero.
BIIC$V_EIVECTOR	12-bit vector used in error interrupt sequences.
<15:14> [1]	Reserved to DIGITAL. Must be zero.
BIIC$V_LEVEL	These four bits (<19:16>) correspond to the four interrupt levels (INT <7:4>) of the VAXBI bus. A set bit causes the corresponding level to be used when INTR commands under control of this register are transmitted.
BIIC$V_EIFORCE	Force bit. When set, this bit posts an error interrupt request in the same way as a bit set in the Bus Error Register (BIIC$L_BER), except that the request is not qualified by the bits BIIC$V_HEIE and BIIC$V_SEIE in BIIC$L_BICSR.
BIIC$V_EISENT[2]	INTR sent.
<22> [1]	Reserved to DIGITAL. Must be zero.
BIIC$V_EIINTC[2]	INTR complete. When set, this bit indicates that the vector for an error interrupt has been successfully transmitted or an INTR command sent under the control of this register has been successfully aborted.
BIIC$V_EIINTAB[2]	INTR abort. When set, this bit indicates that an INTR command under the control of this register has been aborted (that is, a NO ACK or illegal confirmation code has been received). This bit is a status bit set by the BIIC and can be reset only by the user interface.
<31:25> [1]	Reserved to DIGITAL. Must be zero.

BIIC$L_IDR	Interrupt Destination Register. The low-order word of this register indicates which nodes are to be selected by INTR commands.
BIIC$L_IPIMR	Interprocessor Interrupt Mask Register. The high-order word of this register indicates which nodes are permitted to send IPINTRs to this node.
BIIC$L_IPIDR	Force-bit IPINTR/STOP Destination Register. The low-order word of this register indicates which nodes are to be targeted by force-bit IPINTR or STOP commands sent by this node.
BIIC$L_IPISR[2]	IPINTR Source Register. The BIIC stores in the high-order word of this register the decoded ID of a node that sends an IPINTR command to this node.

[1]Read-only field.

[2]Write-one-to-clear bit. Write-type transactions cannot set this bit.

Table B–1 (Cont.) Contents of the BIIC Registers

Field Name	Contents
BIIC$L_SAR	Starting Address Register. The Starting Address Register and Ending Address Register define storage blocks in either memory or I/O space. They must not be configured to include nodespace or multicast space. The low-order 18 bits of this register must be zero. This means that memories are multiples of 256K bytes. Software should set up the Starting Address Register before the Ending Address Register.
BIIC$L_EAR	Ending Address Register. The low-order 18 bits of this register must be zero. This means that memories are multiples of 256K bytes. Software should set up the Starting Address Register before the Ending Address Register. See the description of the Starting Address Register (BIIC$L_SAR) above.
BIIC$L_BCICR	BCI Control Register. The following fields are defined within BIIC$L_BCICR.

Bit Field	Contents
<2:0> [1]	Reserved to DIGITAL. Must be zero.
BIIC$V_RTOEVEN	RTO EV enable.
BIIC$V_PNXTEN	Pipeline NXT enable.
BIIC$V_IPINTREN	IPINTR enable.
BIIC$V_INTREN	INTR enable.
BIIC$V_BICSREN	BIIC CSR Space enable.
BIIC$V_UCSREN	User Interface CSR Space enable.
BIIC$V_WINVALEN	WRITE Invalidate enable.
BIIC$V_INVALEN	INVAL enable.
BIIC$V_IDENT	IDENT enable.
BIIC$V_RESEN	RESERVED enable.
BIIC$V_STOPEN	STOP enable.
BIIC$V_BDCSTEN	BDCST enable.
BIIC$V_MSEN	Multicast Space enable.
BIIC$V_IPINTRF	IPINTR/STOP force.
BIIC$V_BURSTEN	Burst enable.
<31:18> [1]	Reserved to DIGITAL. Must be zero.

[1] Read-only field.

Table B–1 (Cont.) Contents of the BIIC Registers

Field Name	Contents
BIIC$L_WSR	Write Status Register. The following fields are defined within BIIC$L_WSR.

Bit Field	Contents
<27:0> [1]	Reserved to DIGITAL. Must be zero.
BIIC$V_GPR0[2]	Indicates that a VAXBI transaction has written to General Purpose Register 0 (BIIC$L_GPR0).
BIIC$V_GPR1[2]	Indicates that a VAXBI transaction has written to General Purpose Register 1 (BIIC$L_GPR1).
BIIC$V_GPR2[2]	Indicates that a VAXBI transaction has written to General Purpose Register 2 (BIIC$L_GPR2).
BIIC$V_GPR3[2]	Indicates that a VAXBI transaction has written to General Purpose Register 3 (BIIC$L_GPR3).

Field Name	Contents
BIIC$L_IPISTPF	Force-Bit IPINTR/STOP Command Register. The following fields are defined within BIIC$L_IPISTPF.

Bit Field	Contents
<10:0> [1]	Reserved to DIGITAL. Must be zero.
BIIC$V_MIDEN	Master ID Enable.
BIIC$V_CMD	These four bits indicate the command code for either an IPINTR or STOP transaction that is initiated by setting the IPINTR/STOP force bit (BIIC$V_INTRF in BIIC$L_BCICR).
<31:16> [1]	Reserved to DIGITAL. Must be zero.

Field Name	Contents
BIIC$L_UICR	User Interface Interrupt Control Register. This register controls the operation of interrupts initiated by the device. The following fields are defined within BIIC$L_UICR.

Bit Field	Contents
<1:0> [1]	Reserved to DIGITAL. Must be zero.
BIIC$V_UIVECTOR	These 12 bits contain the vector used during user interface interrupt sequences (unless the external vector bit (BIIC$V_EXVECTOR in BIIC$L_UICR) is set). The vector is transmitted when this node wins an IDENT arbitration that matches the conditions given in BIIC$L_UICR.
<14> [1]	Reserved to DIGITAL. Must be zero.

[1]Read-only field.

[2]Write-one-to-clear bit. Write-type transactions cannot set this bit.

Table B–1 (Cont.) Contents of the BIIC Registers

Field Name	Contents
	Bit Field / **Contents**
	BIIC$V_EXVECTOR — When set, the BIIC solicits the interrupt vector from the node rather than transmitting the vector contained in BIIC$L_UICR.
	BIIC$V_UIFORCE — These four bits correspond to the four interrupt levels (INT <7:4>). When a bit is set, the BIIC generates an interrupt at the indicated level.
	BIIC$V_UISENT[2] — These four bits correspond to the four interrupt levels (INT <7:4>). A set bit indicates that an INTR command for the corresponding level has been successfully transmitted.
	BIIC$V_UIINTC[2] — These four bits correspond to the four interrupt levels (INT <7:4>). A set bit indicates that the vector for an interrupt at the corresponding level has been successfully transmitted or that an INTR command sent under the control of this register has been successfully aborted.
	BIIC$V_UIINTAB[2] — These four bits correspond to the four interrupt levels (INT <7:4>). A set bit indicates that an INTR command at the corresponding level, sent under the control of this register, has been aborted (that is, a NO ACK or illegal confirmation code has been received).
BIIC$L_GPR0	General Purpose Register 0
BIIC$L_GPR1	General Purpose Register 1
BIIC$L_GPR2	General Purpose Register 2
BIIC$L_GPR3	General Purpose Register 3

[2]Write-one-to-clear bit. Write-type transactions cannot set this bit.

B.8.2 IOC$ALLOSPT

Drivers for non-DIGITAL-supplied VAXBI device drivers use the executive routine IOC$ALLOSPT when they need to map a portion of a device's nodespace to system virtual address space. See Section B.4.3 for a discussion of a driver's use of IOC$ALLOSPT to map a device's VAXBI window space.

A description page for the routine follows:

IOC$ALLOSPT

Module: IOSUBNPAG

Drivers call IOC$ALLOSPT to allocate a number of entries from the system page table.

input

Registers	Contents
R1	Number of system page table entries to be allocated.
Fields	**Contents**
BOO$GL_SPTFREL	Lowest free virtual page number.
BOO$GL_SPTFREH	Highest free virtual page number.

IPL at execution: IPL$_SYNCH

output

Registers	Contents
R0	SS$_NORMAL or 0
R1	Number of allocated system page table entries.
R2	Starting system virtual page number (SVPN) allocated.
R3	Address of the base of the system page table (MMG$GL_SPTBASE).

IPL at exit: IPL$_SYNCH

Index

A

Access control list • 3-13
Adapter • B-2
Adapter control block
  See ADP
Address space
  See VAXBI address space
ADP (adapter control block)
  for generic VAXBI devices • B-8 to B-9
ADP$L_BI_IDR • B-8, B-12
ADP$W_BI_VECTOR • B-8, B-12
Arbitration mode • B-22
AST routine
  invoked by screen management routine • 3-15
AT$_GENBI • B-8
AUTOCONFIGURE command • B-19
AUTOGEN • 1-3, 1-15
  calculations • 2-7

B

Backplane interconnect interface chip
  See BIIC
Backup of console disk • 1-6, 1-11
Backup of system disk • 1-6
BIIC (backplane interconnect interface chip) • B-6
  clearing error register • B-12
  enabling error interrupts • B-13, B-23
  self test • B-11
BIIC$L_BCICR • B-13, B-25
BIIC$L_BER • B-7, B-12, B-13, B-22 to B-23
BIIC$L_BICSR • B-11, B-22
BIIC$L_DTREG • B-6, B-22
BIIC$L_EAR • B-24
BIIC$L_EICR • B-9, B-12, B-23 to B-24
BIIC$L_GPR0 • B-27
BIIC$L_GPR1 • B-27
BIIC$L_GPR2 • B-27
BIIC$L_GPR3 • B-27
BIIC$L_IDR • B-12, B-24
BIIC$L_IPIDR • B-24
BIIC$L_IPIMR • B-24
BIIC$L_IPISR • B-24
BIIC$L_IPISTPF • B-26
BIIC$L_SAR • B-24
BIIC$L_UICR • B-9, B-12, B-26 to B-27
BIIC$L_WSR • B-26
BIIC$V_BROKE • B-11
BIIC$V_SST • B-11
BIIC$V_STS • B-11
BIIC CSR space • B-6
$BIICDEF macro • B-6, B-21
BIIC registers
  accessing • B-6
  symbolic names • B-21 to B-27
Booting
  with XDELTA • 3-19
Bootstrap command procedure
  alternate nonstop • 2-4
Bootstrapping
  with volume shadowing • 2-4
BUA (BI-to-UNIBUS adapter) • B-9
Buffered I/O • B-16

C

Channel request block
  See CRB
CI–780 • 2-6
CI port driver (PADRIVER) • 2-6 to 2-7
Common system root • 1-2, 1-3
Concurrent update • 1-3, 1-5
CONFREGL array • B-7
CONNECT command • B-20
Controller initialization routine • B-10
  forking • B-15
CRB (channel request block)
  for generic VAXBI devices • B-8

D

Debugger • 3-16
DECnet–VAX
  ULTRIX-32 File Operations • 3-17
$DEF macro • 3-20
Delta/XDelta Utility (DELTA/XDELTA) • 3-19
DELUA communications controller
  with broadband • 3-14

DEQNA communications controller
with DECOM transceiver • 3-14
DEUNA communications controller
with broadband • 3-14
Device • B-2
"offsettable" • B-9
Device initialization
performed by VAX/VMS • B-6 to B-9
performed by VAXBI device driver • B-10 to B-15
Device registers
accessing • B-6
Direct I/O • B-16
Diskette device
MSCP-served • 3-7
Disk space
freeing • 1-14
DMA (direct memory access) transfer • B-15 to B-19
DMB32 asynchronous/synchronous multiplexer • B-17
DSDRIVER.EXE • 1-2
DSDRIVER.V44EXE • 1-2
DTE
state transitions • 3-10
DTE states • 3-9
DXCOPY command procedure • 2-2, 2-3
Dynamic image setting • 3-16

E

ETDRIVER.EXE • 2-8
Ethernet communications controller
expected errors • 3-14
promiscuous mode restriction on driver code • 3-14
VAXBI • 2-8, 3-13
EXE$ALLOCBUF • B-16
EXE$ALOPHYCNTG • B-18
EXE$GL_CONFREGL • B-7
EXE$QIODRVPKT • 3-19

F

FDT (function-decision table) routine • B-16
Fixed CSR space
of non-DIGITAL-supplied devices • 2-9
Fixed vector space
of non-DIGITAL-supplied devices • 2-9
Forking • B-15

G

Global pages
required for installation • 1-8 to 1-9
Global sections
required for installation • 1-8 to 1-9

I

IDB (interrupt dispatch block)
for generic VAXBI devices • B-8
IDB$L_UCBLST • B-20
IEEE 802 formatted packets • 3-14
IFNORD macro • 3-18
IFNOWRT macro • 3-18
IFRD macro • 3-18
IFWRT macro • 3-18
INIADP module • B-6 to B-9
Installation • 1-1 to 1-17
completing • 1-14 to 1-16
from diskette • 1-12
from tape • 1-12
global pages required • 1-8 to 1-9
global sections required • 1-8 to 1-9
making space for • 1-7
messages • 1-13
of optional software • 3-6
on VAXcluster • 1-2 to 1-5, 1-15
performing • 1-10 to 1-14
preparing the system for • 1-6 to 1-10
required limits for SYSTEM account • 1-7 to 1-8
selecting an update option • 1-12
specifying a device name • 1-11
Interrupt destination
setting • B-12
Interrupt dispatch block
See IDB
Interrupt dispatcher • B-9
Interrupt vector
for VAX 8200 and VAX 8300 • B-9
for VAX 8500, VAX 8550, VAX 8700, and VAX 8800 • B-9
setting • B-12
IOC$ALLOSPT • B-14, B-27 to B-28
IOC$MOVFRUSER • B-18
IOC$MOVTOUSER • B-19

J

Journal file • 1-17

L

LINK/SHARE/DEBUG command • 3-16
LINK/SHARE/NOTRACE command • 3-16
Linker
  shareable image • 3-16
LKI$_BLOCKEDBY item code • 3-12
LKI$_BLOCKING item code • 3-12
LKI$_LOCKS item code • 3-12
Logical name table
  process-private • 3-17

M

MAKEROOT command procedure • 1-3
Memory
  See VAXBI memory
MMG$GL_SBICONF • B-7
MONITOR server processes
  permanent • 3-5
MSCP server
  restriction on diskette devices • 3-7
Multiprocessing • 1-2

N

NETACP (network ancillary control program)
  verification of MOP messages • 3-18
Net disk block utilization • 1-7, 1-14
Network
  shutting down • 1-9
NMA$C_PCLI_FMT parameter • 3-14
NMA$C_PCLI_PAD parameter • 3-14
NMA$C_PCLI_PRM parameter • 3-14
NMI-to-BI adapter • B-2
Node
  See VAXBI node
Nodespace • B-3, B-6
  mapped by VAX/VMS • B-7
NOP (No Operation) instruction
  restriction in use • 3-16

O

Optional software • 3-6

P

PADRIVER.EXE • 2-6 to 2-7
Page table
  physical address of • B-18
Page table entry
  format • B-17
Patch
  printing Version 4.5 listing • 1-17
Peak disk block utilization • 1-7
Private system root • 1-2
Promiscuous mode • 3-14
$PRTCTEND macro • B-11
$PRTCTINI macro • B-11

R

Register dumping routine • B-19
Registers
  See Device registers, BIIC registers
Remote terminal
  error count • 3-3
Rolling update • 1-3 to 1-4

S

SBICONF array • B-7
SCAN
  VARYING STRING input stream • 3-15
SCB (system control block)
  of VAX 8200 and VAX 8300 • B-9
  of VAX 8500, VAX 8550, VAX 8700, and VAX 8800 • B-9
SCNRTL • 3-15
Security auditing information
  protection • 3-8 to 3-9
Self-test status • B-22
  determining • B-11
SET (WIDTH,...) built-in procedure • 3-2
SET SCREEN command • 3-2

Shareable image
traceback imformation passed to • 3-16
SHOW DEVICE command • 3-3
SHOW MEMORY command
for VAX 8300 • 1-16
Small-disk system • 1-12
SMG$ENABLE_UNSOLICITED_INPUT • 3-15
SMG$SET_BROADCAST_TRAPPING • 3-15
SMG$SET_OUT_OF_BAND_ASTS • 3-15
Spooled line printer
setting up queue • 3-5
Standalone BACKUP • 1-6
error reported while booting • 3-13
rebuilding • 1-15
STARTUP.COM • B-19
SUBMIT/AFTER command
behavior in VAXcluster • 3-1
SYS$CREMBX • 3-17
SYS$GETLKI
invalid returned length address field in • 3-12
SYS$QIO
zero-length transfer • 3-19
SYS$SETIMR • 3-16
SYS$UPDATE:VMS045.TXT • 1-12, 1-17, A-1 to A-32
SYS$WAITFR • 3-16
SYSGEN
See System Generation Utility
SYSTEM account
required limits • 1-7 to 1-8
System control block
See SCB
System Generation Utility (SYSGEN) • 2-7
device table • 2-9 to 2-14
loading a VAXBI device driver • B-19 to B-20
System page table entry
allocating • B-14
System root • 1-2

T

Tailored system • 1-12
installing layered products on • 3-3 to 3-5
update journal file • 1-17
Terminal
error count • 3-3
TFDRIVER.EXE • 3-15
TIMEWAIT macro • 3-16
Traceback
information passed to shareable image • 3-16
TU78 Tape Driver (TFDRIVER.EXE) • 3-15

U

UCB$L_SVAPTE • B-16
UCB$V_ONLINE • B-11
UCB$W_BCNT • B-16
UCB$W_BOFF • B-16
UETP (User Environment Test Package) • 2-4 to 2-5
UETUNAS00.EXE • 2-4
ULTRIX-32
File Operations with DECnet–VAX • 3-17
Unit initialization routine • B-10
forking • B-15
Update
disk block utilization • 1-7
for tailored system • 1-12
media • 1-1
options • 1-12
Update description file • 1-12, 1-17, A-1 to A-32
Update kit
contents • 1-1
contents • A-1 to A-32
User Environment Test Package
See UETP
User interface CSR space
enabling interrupts from • B-13

V

VAX-11/725 • 1-1
VAX-11/730 • 1-1
VAX-11/780 • 1-1
VAX-11/782 • 1-1
VAX-11/785 • 1-1
VAX 8200 • 1-1, B-1
alternate nonstop bootstrap • 2-4
bootstrapping from HSC-controlled disk • 2-1 to 2-3
interrupt destination • B-8
VAX 8300 • B-1
alternate nonstop bootstrap • 2-4
bootstrapping from HSC-controlled disk • 2-1 to 2-3
interrupt destination • B-8
32MB memory system • 1-16
multiprocessing • 1-2
VAX 8500 • B-1
booting with XDELTA • 3-19
bootstrapping with volume shadowing • 2-4

VAX 8500 (cont'd.)
interrupt destination • B-8
NOP instruction restriction • 3-16
VAX 8550 • B-1
booting with XDELTA • 3-19
bootstrapping with volume shadowing • 2-4
interrupt destination • B-8
NOP instruction restriction • 3-16
VAX 8600 • 1-1, 1-2, 1-6
NOP instruction restriction • 3-16
VAX 8650 • 1-1, 1-2, 1-6
NOP instruction restriction • 3-16
VAX 8700 • B-1
booting with XDELTA • 3-19
bootstrapping with volume shadowing • 2-4
interrupt destination • B-8
NOP instruction restriction • 3-16
VAX 8800 • 1-1, B-1
bootstrapping with volume shadowing • 2-4
interrupt destination • B-8
multiprocessing • 1-2
NOP instruction restriction • 3-16
VAX BASIC
SET INITIAL CHOICE statement • 3-17
VAXBI
address space • B-3 to B-6
device support • B-1 to B-28
errors • B-22
I/O space • B-3
memory • B-3
node • B-2, B-6
VAXBI port communications controller • 2-8, 3-13
VAXcluster
applying update to • 1-2 to 1-5
behavior of SUBMIT/AFTER command • 3-1
copying VMB.EXE to members' console media • 1-15
performing a concurrent update • 1-5
performing a rolling update • 1-3 to 1-4
VAX PSI
DTE state transitions • 3-10
VAX SCAN
PRUNE statement • 3-15
SCN$GET_TOKEN_NAME • 3-15
VAX Text Processing Utility
See VAXTPU
VAXTPU (VAX Text Processing Utility) • 3-2
GET_INFO built-in procedure • 3-2
VMSINSTAL
alternate mode • 1-7
automatic purge • 1-13
installing optional products using • 3-6
VMSINSTAL (cont'd.)
selecting an update option • 1-12
VMSUPDATE
restrictions on use • 3-6
Volume shadowing
bootstrapping with • 2-4
installing the product key • 1-1

W

Window space • B-6
mapping • B-13 to B-14

X

X.25 packet level events • 3-5
XABPRO block
XAB$L_ACLSTS field • 3-13

READER'S COMMENTS

Note: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Did you find this manual understandable, usable, and well organized? Please make suggestions for improvement.

_____

_____

_____

_____

_____

_____

_____

_____

_____

Did you find errors in this manual? If so, specify the error and the page number.

_____

_____

_____

_____

_____

_____

_____

_____

_____

Please indicate the type of user/reader that you most nearly represent:

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Other (please specify) _____

Name _____ Date _____

Organization _____

Street _____

City _____ State _____ Zip Code _____
or Country

— — — Do Not Tear - Fold Here and Tape —

digital

No Postage
Necessary
if Mailed in the
United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO.33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

SSG PUBLICATIONS ZK1-3/J35
DIGITAL EQUIPMENT CORPORATION
110 SPIT BROOK ROAD
NASHUA, NEW HAMPSHIRE 03062-2698

— — — Do Not Tear - Fold Here —

Cut Along Dotted Line

READER'S COMMENTS

Note: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Did you find this manual understandable, usable, and well organized? Please make suggestions for improvement.

_____

_____

_____

_____

_____

_____

_____

_____

_____

Did you find errors in this manual? If so, specify the error and the page number.

_____

_____

_____

_____

_____

_____

_____

_____

_____

Please indicate the type of user/reader that you most nearly represent:

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Other (please specify) _____

Name _____ Date _____

Organization _____

Street _____

City _____ State _____ Zip Code_____
or Country

— — — Do Not Tear - Fold Here and Tape —

digital

No Postage
Necessary
if Mailed in the
United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO.33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

SSG PUBLICATIONS ZK1-3/J35
DIGITAL EQUIPMENT CORPORATION
110 SPIT BROOK ROAD
NASHUA, NEW HAMPSHIRE 03062-2698

— — — Do Not Tear - Fold Here —

Cut Along Dotted Line